

# XVT-Design++<sup>TM</sup>

*AN INTERACTIVE C++ APPLICATION FRAMEWORK FROM THE #1 COMPANY IN MULTI-PLATFORM DEVELOPMENT TOOLS.*



XVT

XVT-Design++

Volume

2



# **XVT++ 2.0 CLASS LIBRARY REFERENCE**

---

**XVT - THE PORTABLE GUI DEVELOPMENT SOLUTION**

---





## Copyrights

© 1993 XVT Software Inc. All rights reserved.

The XVT++ application program interface, XVT manuals and technical literature may not be reproduced in any form or by any means except by permission in writing from XVT Software Inc.

XVT and XVT++ are trademarks of XVT Software Inc. Other product names mentioned in this document are trademarks or registered trademarks of their respective holders.

## Published By

XVT Software Inc.  
Box 18750  
Boulder, CO 80308  
(303) 443-4223  
(303) 443-0969 (fax)

## Credits

**Original Concept and Design**  
Marc Rochkind

**Architecture and Specification**  
Scott Meyer and Jack Unrue

**Engineers**  
Scott Meyer, Jack Unrue, Steve Archuleta

**Documentation**  
Lynn Merrill, Scott Meyer, Dave Welsch

**Product Team**  
Dave Locke, Scott Meyer, Peggy Reed, Connie Leserman,  
Catherine Connor, Jonathan Auerbach

## Printing History

First printing .....March, 1993 ..... XVT++ Version 2.0  
Second printing .....June, 1993 ..... XVT++ Version 2.0

This manual was printed on recycled paper  
by Continental Graphics, Broomfield, Colorado.





# XVT++

---

## CONTENTS

<b>PREFACE .....</b>	<b>v</b>
How to Use This Manual.....	v
XVT_Class .....	vi
<b>INTRODUCTION TO XVT++ .....</b>	<b>1</b>
Overview .....	1
XVT++ and the Interactive Design Tool.....	1
Usage .....	1
Handlers.....	2
Compatibility .....	3
Class Hierarchy .....	3
<b>XVT++ CLASSES .....</b>	<b>5</b>
XVT_AllocErrorHandler.....	5
XVT_AllocErrorManager .....	8
XVT_Base .....	10
XVT_BaseDrawProto .....	15
XVT_Brush .....	37
XVT_Button .....	41
XVT_CheckBox .....	44
XVT_ChildBase .....	48
XVT_ChildWin .....	66
XVT_ClipBoard .....	72
XVT_Color .....	80
XVT_Config .....	84
XVT_Container .....	90
XVT_Control .....	92
XVT_DetachedWin .....	101
XVT_Dialog .....	108



XVT_Directory .....	125
XVT_DrawableContainer .....	127
XVT_DrawTools .....	150
XVT_Edit .....	158
XVT_Editable .....	161
XVT_ErrorHandler.....	166
XVT_ErrorManager .....	169
XVT_FileSpec .....	171
XVT_Font .....	175
XVT_FontMetrics .....	178
XVT_GlobalAPI .....	182
XVT_GroupBox .....	226
XVT_Icon .....	229
XVT_InternalErrorHandler .....	233
XVT_InternalErrorManager.....	236
XVT_Label .....	238
XVT_List .....	243
XVT_ListBox .....	253
XVT_ListButton .....	258
XVT_ListEdit .....	262
XVT_Menu .....	267
XVT_MenuItem .....	273
XVT_MenuNode .....	277
XVT_MenuNodeBase .....	281
XVT_MenuSeparator .....	283
XVT_MenuWin .....	285
XVT_Pen .....	295
XVT_Picture .....	300
XVT_Pnt .....	306
XVT_PrintWin .....	309
XVT_RadioButton .....	318
XVT_Rct .....	323
XVT_ScreenWin .....	333
XVT_ScrollBar .....	337
XVT_StaticText .....	346
XVT_StrList .....	349
XVT_SubMenu .....	356
XVT_TaskWin .....	359
XVT_TextEdit .....	370
XVT_Timer .....	390
XVT_Toggle .....	393
XVT_TopLevelWin .....	397

<b>XVT++ 1.1 COMPATIBILITY CLASSES.....</b>	<b>407</b>
XVT++ 1.1 to 2.0 Member Function Map .....	407
BaseWin .....	414
BaseXVT .....	424
Brush .....	432
Control .....	436
DlgWin .....	453
DrawTools .....	459
Font .....	463
GraphWin .....	468
MenuItem .....	482
Pen .....	486
Pnt .....	490
Rct .....	493
ScreenWin .....	498
StrList .....	509
StrListElt .....	517
TaskWin .....	518
<b>INDEX OF EQUIVALENT C FUNCTIONS .....</b>	<b>523</b>
<b>GENERAL INDEX .....</b>	<b>529</b>







---

# PREFACE

## How to Use This Manual

This reference manual lists class names in alphabetical order. Each class description follows the same format. If any section of the class description is not used for a particular class, it is omitted.

The following pages of this preface illustrate and describe the format. In addition, several typesetting conventions indicate different types of information:

*code*

This typestyle represents code, including expressions and the names of functions, attributes, variables, and structures.

### **file names**

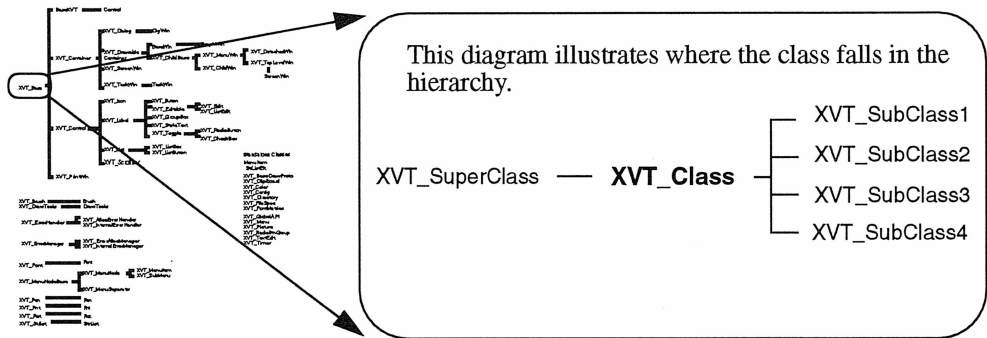
Bold type is used for file names and extensions.

*emphasis*

Italics are used for emphasis and for the names of documents.

The following pages illustrate the class description format.

# XVT\_Class



## Overview

Header File	The header file in which the class definition for this class can be found. You should not include this file directly; always use <b>xvtp.h</b> to include XVT++ definitions in your application.
Source File	The source file in which the code for the class member functions is found.
Superclass	The superclass from which this class is derived. Since XVT++ does not employ multiple inheritance, each class has only one superclass.
Subclasses	Subclasses of this class.
Usage	One of the following: <b>Abstract:</b> You may create only subclasses of this class. <b>Concrete:</b> You may create both instances and subclasses of this class. <b>Implementation:</b> This class is part of the implementation of XVT++ and should be neither subclassed nor instantiated.

An overview of the class, what it does, and how to use it appears after the summary table.

## Example

An example of how to use this class.

## Constructors

Descriptions of class constructors and destructors; listed only for concrete and abstract classes, not for implementation classes.

## Operators

A listing of any operators overloaded by this class.

## Casts

A listing of virtual cast functions implemented by this class.

## Member Functions

This section describes class member functions. All member functions described constitute the documented XVT++ interface. Some undocumented member functions are part of the XVT++ implementation. Use undocumented functions at your own risk; they may not work the way you expect, and they may change or disappear entirely from future releases.

Private member functions are always part of the implementation.

In some cases, subclasses override member functions provided by a superclass without changing the syntax or semantics of the member function. This is done to get around the fact that operator overloading only applies to functions in the same class, or to take advantage of pure virtual functions. In these cases, the override is not documented in the subclass, only in the superclass.

---

## XVT\_Class::FunctionName

A SHORT DESCRIPTION OF THE MEMBER FUNCTION

---

### Prototypes

The member function prototypes.

### Parameters

A description of each parameter. In the case of overloaded functions, every parameter with the same name has the same semantics and is thus described only once.

### Return Value

A description of the function's return value.

### Description

A general description of the member function.

an overloaded member function

A description of a particular overloaded member function.

### Implementation Notes

Any notes on platform dependencies relating to this function.

### Equivalent C Functions

A list of equivalent C functions from the XVT Portability Toolkit.

## Implementation Members

A list of member functions and variables that are part of the implementation but that are not part of the interface. Do not use implementation members in your code; implementation members will not necessarily remain the same for future releases of XVT++.

## Inherited Member Functions

A list of inherited member functions.

### From XVT\_SuperClass

*page number*      function prototype



**Reference**

# 1

---

## INTRODUCTION TO XVT++

### Overview

XVT++ 2.0 provides a complete C++ interface to the functionality offered by the XVT Portability Toolkit.

### XVT++ and the Interactive Design Tool

This product is designed to be used with the Interactive Design Tool (IDT). Together, the IDT and the XVT++ Class Library are called XVT-Design++. We do not recommend writing XVT++ applications without the IDT, even though it is possible (though more difficult) to do so.

XVT strongly suggests that *new* applications be produced using the IDT. We believe this is the most productive way to use our products, and XVT is best able to provide customer support using this method of application development.

### Usage

A typical XVT++ application consists of a subclass of `XVT_TaskWin` and one or more subclasses of the GUI container classes: `XVT_ToplevelWin`, `XVT_DetachedWin`, and `XVT_Dialog`. Each subclass overrides whichever event handler member functions are necessary for the application to function. With the exception of menus, all GUI objects have at least `e_create` and `e_destroy` event handler member functions, which are called when the object is created and just before it is destroyed, respectively. All GUI objects, including menus, have constructor and destructor member functions. These are called when the C++ object is constructed and destroyed.

Any of the GUI container classes can contain controls and, in the case of windows, text edit objects and child windows. Like GUI containers, controls have their own event handler member functions that are overridden by user subclasses. Most controls have at least `e_create`, `e_destroy`, and `e_action` event handler member functions.

All GUI objects—controls, windows, and dialogs—have a two-phase creation protocol. The two-phase protocol prevents a problem that can occur when the window system causes recursion in a C++ constructor: callbacks from the window system can cause the application program to try to use an object that is not yet completely constructed. In the two-phase protocol, the GUI object is first created with the C++ `new` operator and then initialized with the `Init` member function. The object's `e_create` member function is called before `Init` returns.

## Handlers

XVT++ 2.0 preserves the style of programming used in version 1.1: you are expected to create subclasses that override virtual event handler member functions (the `e_*` functions) to implement whatever behavior your application needs. For many applications, this scheme is satisfactory; however, there are times when other techniques are preferred.

One such situation is the case where an application has many controls or windows that are very similar, for example 50 text entry fields that collect data for a database query. Creating 50 distinct classes results in much duplicate code. A better solution is to create a single class that changes its behavior based on parameters provided in the constructor, in additional member functions, or in resource user data. You can then create 50 instances of this class, one for each control.

In other cases, the fact that all of an object's behavior must be specified in a single subclass definition causes difficulties. A symptom of this sort of problem would be subclass member functions that all start with `if` or `switch` statements, which cause the member function to behave in completely different ways based on the state of the object.

A cleaner approach is to create what are known as behavior or delegate objects. A behavior object implements a single type of behavior; it has no `switch` statements. The behavior of an object is

changed by replacing behavior objects instead of taking different paths through switch logic.

**Tip:** The simplest way to implement behaviors is to create an abstract behavior class that has member functions corresponding to the event handling member functions present in the XVT++ object. Each actual behavior will be a subclass of this behavior class.

The subclass of the XVT++ object is very simple. It adds storage for a current behavior pointer and implementations of the event handling member functions that just call the corresponding function in the current behavior. When the subclass is instantiated, it installs the behavior corresponding to the start state. As the object is manipulated, the current behavior is called and can manipulate the XVT object subclass as required, including switching the current behavior.

## Compatibility

XVT++ 2.0 is fully backwards compatible with XVT++ 1.1. Most XVT++ 1.1 programs should run without modification. The exception to this rule is XVT++ 1.1 programs that rely on specific details of the 1.1 inheritance hierarchy, for example that both `DlgWin` and `ScreenWin` are subclasses of `BaseWin`. To give 1.1 applications full access to the new functionality provided by 2.0, it was necessary to have the old 1.1 classes inherit from the new 2.0 classes. The alternative, leaving the 1.1 hierarchy alone, would effectively isolate 1.1 applications from the new features.

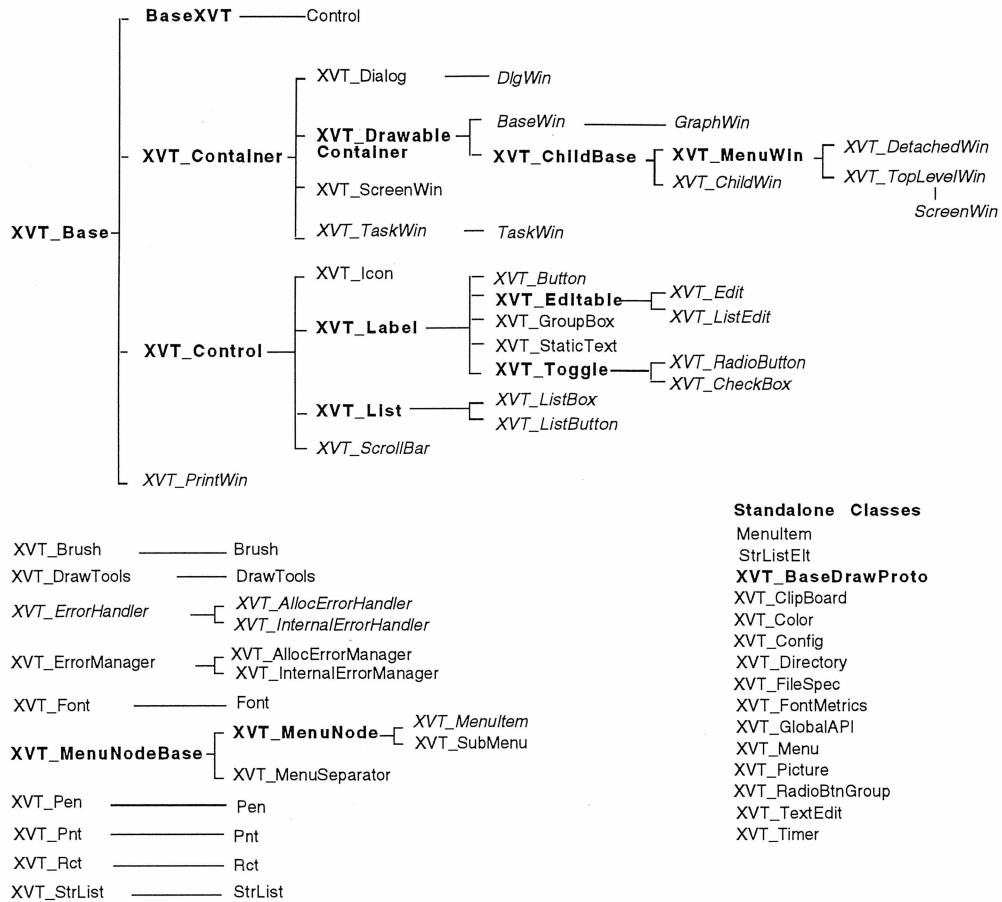
The impact of this decision would be noticed by applications that rely on the polymorphism provided by the old hierarchy, by having a list of `BaseWins`, for example. Since the operations defined by `BaseWin` are no longer inherited by `DlgWin` or `ScreenWin` (they are re-implemented in those classes), you can't cast a `ScreenWin` to a `BaseWin` or vice-versa.

However, you can cast both `ScreenWin` and `BaseWin` to a common ancestor, the `XVT_DrawableContainer` class. By changing the list of `BaseWins` to a list of `XVT_DrawableContainers`, you can maintain the polymorphism allowed by the old hierarchy. The dynamic downcast routines provided by `XVT_Base` allow for safe downcasting so that the original `BaseWin` code can be used.

## Class Hierarchy

Figure 1 shows a diagram of the XVT++ class hierarchy.





**Key**

Concrete Class

Abstract Class

Implementation Class

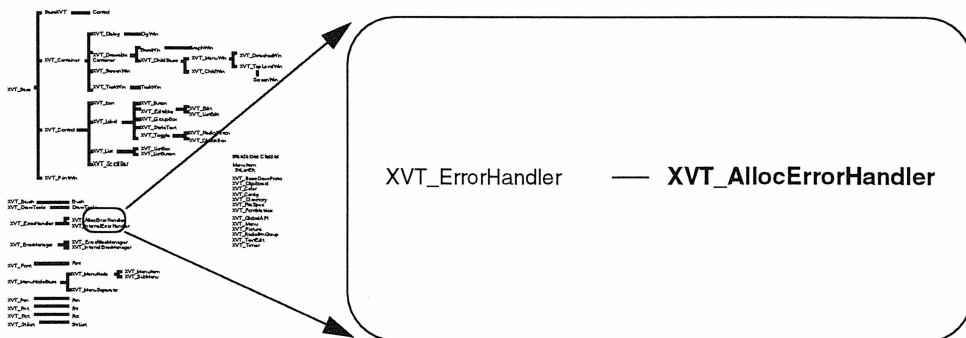
Figure 1: The XVT++ Class Hierarchy.

## 2

## XVT++ CLASSES

This chapter describes the XVT++ classes, except for the 1.1 compatibility classes, which are described in Chapter 3.

## XVT\_AllocErrorHandler



## Overview

<b>Header File</b>	<code>error.h</code>
<b>Source File</b>	<code>error.cc</code>
<b>Superclass</b>	<code>XVT_ErrorHandler</code>
<b>Subclasses</b>	
<b>Usage</b>	Abstract

This class defines the interface to all memory allocation error handlers. To create your own memory allocation error handler, you would create a subclass that provides an implementation of `Handler`, which takes whatever recovery actions you want.

## Constructors

`XVT_AllocErrorHandler()`

## Member Functions

---

### XVT\_AllocErrorHandler::Handler

HANDLE A MEMORY ALLOCATION ERROR

---

#### Prototypes

protected:

```
virtual BOOLEAN  
Handler() = 0
```

#### Return Value

TRUE if the handler resolved the error condition and program execution can continue, FALSE if the next handler in the chain should be tried.

#### Description

This function is called by `Handle` when this error handler is given a chance to handle a memory allocation error. Your subclass must provide an implementation that takes whatever recovery actions are necessary.

A typical strategy for handling memory allocation errors is to allocate a substantial amount of memory (say 20K) at program start up and then to free it from within an allocation error handler. If the intent of your implementation is to actually handle the memory allocation error, then you should free up as much memory as you can and return TRUE.

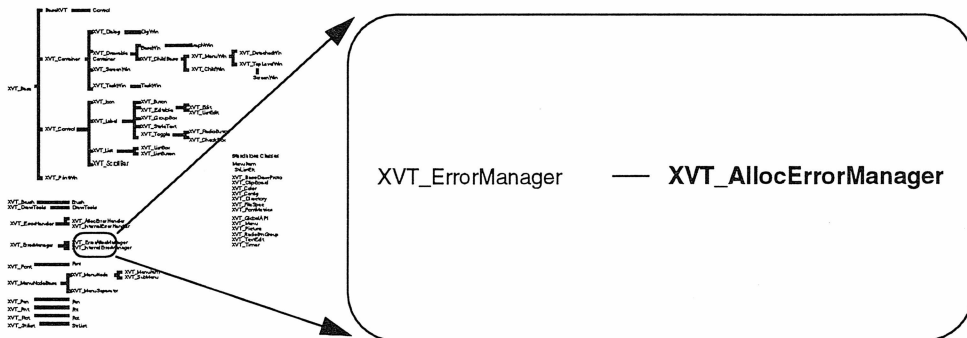
XVT++ uses both `malloc` and `new`. It is not required that `new` and `malloc` use the same heap. Unless they do in fact use the same heap in your target environments, you will have to both `free` and `delete` memory in order to recover reliably.

## Inherited Member Functions

### From XVT\_ErrorHandler

*page 167*      `virtual BOOLEAN Handle( long data )`

# XVT\_AllocErrorManager



## Overview

<b>Header File</b>	error.h
<b>Source File</b>	error.c
<b>Superclass</b>	XVT_ErrorManager
<b>Subclasses</b>	
<b>Usage</b>	Concrete

Instances of this class handle memory allocation errors. These errors arise either when new fails or when the underlying XVT toolkit is not able to allocate more memory.

There is only one instance of this class, pointed to by the global variable, XVT\_AllocError.

## Constructors

XVT\_AllocErrorManager()



## Member Functions

---

### XVT\_AllocErrorManager::Raise

RAISE A MEMORY ALLOCATION ERROR

---

#### Prototypes

```
void  
Raise()
```

#### Return Value

If `Raise` returns, a handler has repaired the out-of-memory condition by releasing memory. The operation that ran out of memory should be retried.

#### Description

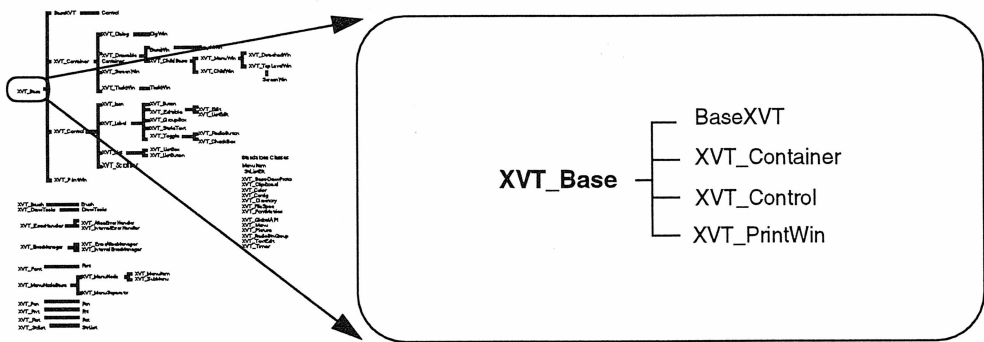
This function is called from two places: the new handler and the error handler registered under `ATTR_MALLOC_ERR_HANDLER`.

## Inherited Member Functions

#### From XVT\_ErrorManager

*page 170*    `virtual void Raise( long data )`

# XVT\_Base



## Overview

Header File	xvtbase.h
Source File	xvtbase.cc
Superclass	
Subclasses	BaseXVT, XVT_Container, XVT_Control, XVT_PrintWin
Usage	Implementation

The XVT\_Base class defines the interface common to all GUI objects that have visible representations on the screen.

## Casts

Virtual cast functions are provided to allow type-safe downcasting. The default implementation of each cast function is to return NULL. Each subclass overrides the corresponding cast function to return a pointer to this instead.

```
virtual DlgWin* CastToDlgWin()
virtual ScreenWin* CastToScreenWin11()
virtual TaskWin* CastToTaskWin11()
```

```

virtual BaseWin* CastToBaseWin()
virtual XVT_Button* CastToButton()
virtual XVT_CheckBox* CastToCheckBox()
virtual XVT_ChildWin* CastToChildWin()
virtual XVT_DetachedWin* CastToDetachedWin()
virtual XVT_Dialog* CastToDialog()
virtual XVT_DrawableContainer* CastToDrawableContainer()
virtual XVT_Edit* CastToEdit()
virtual XVT_GroupBox* CastToGroupBox()
virtual XVT_Icon* CastToIcon()
virtual XVT_ListBox* CastToListBox()
virtual XVT_ListButton* CastToListButton()
virtual XVT_ListEdit* CastToListEdit()
virtual XVT_MenuWin* CastToMenuWin()
virtual XVT_PrintWin* CastToPrintWin()
virtual XVT_RadioButton* CastToRadioButton()
virtual XVT_ScreenWin* CastToScreenWin()
virtual XVT_ScrollBar* CastToScrollBar()
virtual XVT_StaticText* CastToStaticText()
virtual XVT_TaskWin* CastToTaskWin()
virtual XVT_TopLevelWin* CastToTopLevelWin()

```

## Member Variables

---

### XVT\_Base::\_ScreenWin

THE SCREEN WINDOW

---

#### Declaration

```
static XVT_ScreenWin* _ScreenWin;
```

#### Description

A pointer to the screen window.

---

## XVT\_Base::\_TaskWin

THE TASK WINDOW

---

### Declaration

```
static XVT_TaskWin* _TaskWin;
```

### Description

A pointer to the task window.

## Member Functions

---

### XVT\_Base::GetInnerRect

RETRIEVE THE BOUNDARY OF THE CLIENT AREA

---

#### Prototypes

```
virtual XVT_Rct  
GetInnerRect() const
```

#### Return Value

The coordinates of the client area relative to the parent window. This rectangle is *not* normalized, in other words, the upper-left point is not necessarily (0,0). To normalize a rectangle, use the `XVT_Rct::Normalize` member function.

#### Description

Retrieves the boundary of the client area.

For windows and dialogs, the client area is the rectangular area inside the border.

For drop-down controls, the client area is considered to be the size of the control when not dropped down.

For all other types of controls, the client area is identical to the outer boundary.

#### Equivalent C Function

```
get_client_rect()
```

---

## XVT\_Base::GetOuterRect

RETRIEVE THE OUTER BOUNDARY OF ANY GUI OBJECT

---

### Prototypes

```
virtual XVT_Rct  
GetOuterRect() const
```

### Return Value

The coordinates of the outer boundary relative to the parent window.

### Description

Gets the current outer boundary of the object.

The outer boundary is the maximum extent of marks made on the screen by the rendering of the object.

For windows and dialogs, the outer boundary includes the border and any border decorations.

For drop-down controls, the outer boundary is the boundary of the control when dropped.

For all other types of controls, the outer boundary is always identical to what was set with `SetInnerRect`.

### Equivalent C Function

```
get_outer_rect()
```

---

## XVT\_Base::GetType

RETRIEVE THE WINDOW TYPE OF ANY OBJECT

---

### Prototypes

```
virtual WIN_TYPE  
GetType() const
```

### Return Value

The object's window type.

**Description**

Retrieve the window type of this object.

**Equivalent C Function**

`get_window_type`

**Implementation Members**

`XVT_Base`

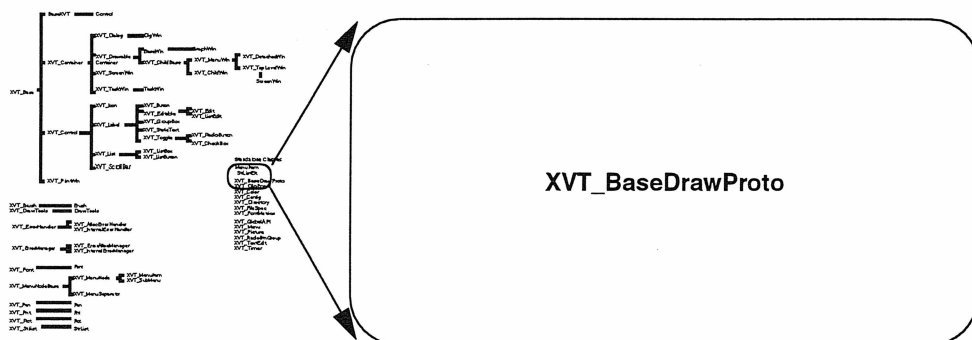
`~XVT_Base`

`GetWindowID`

`WindowID`

`InitProtocols`

## XVT\_BaseDrawProto



## Overview

<b>Header File</b>	<code>draw_p.h</code>
<b>Source File</b>	<code>draw_p.cc</code>
<b>Superclass</b>	
<b>Subclasses</b>	
<b>Usage</b>	Implementation

The `XVT_BaseDrawProto` (Base Drawing Protocol) class provides indirect access to drawing functionality.

Instances of this class are created automatically by objects that can do drawing.

## Member Functions

---

### XVT\_BaseDrawProto::DrawALine

DRAW A LINE

---

#### Prototypes

```
void
DrawALine(
    XVT_Pnt,           point
    BOOLEAN,           start_arrow
    BOOLEAN            end_arrow )
```

#### Parameters

**point**  
The point to draw to.

**start\_arrow**  
A flag that is TRUE if there should be an arrow at the beginning of the line, FALSE if not.

**end\_arrow**  
A flag that is TRUE if there should be an arrow at the end of the line, FALSE if not.

#### Description

Draws a line from the current pen position to point relative to the window's client area.

The pen position for subsequent drawing functions becomes point.

#### Implementation Notes

XVT/CH  
Only horizontal and vertical lines are drawn accurately.

#### Equivalent C Function

win\_draw\_aline()



---

## XVT\_BaseDrawProto::DrawArc

DRAW AN OVAL ARC

---

### Prototypes

```
void  
DrawArc(  
    XVT_Rct          boundary,  
    XVT_Pnt          start  
    XVT_Pnt          stop )
```

### Parameters

**boundary**  
The bounding rectangle for the oval on which the arc is drawn.  
The bounding rectangle should not be empty.

**start**  
Start vector.

**stop**  
Stop vector.

### Description

This function draws an arc that is a section of the perimeter of an oval bounded by **boundary** in the client area of the window. The arc is drawn counter-clockwise along the oval, from the intersection of the start vector and the oval to the intersection of the stop vector and the oval.

### Implementation Notes

XVT/CH  
The rectangle given by **boundary** is drawn instead of an arc.

### Equivalent C Function

`win_draw_arc()`

---

## XVT\_BaseDrawProto::DrawIcon

---

DRAW AN ICON

---

### Prototypes

```
void  
DrawIcon(  
    XVT_Pnt      point  
    long         rid )
```

### Parameters

point  
 Coordinate of the icon's upper left corner.

rid  
 Resource ID.

### Description

This function draws the icon whose resource ID is `rid` so that its upper left corner is at point `(x, y)` in the window's client area. The current background and foreground colors are used. The current drawing mode, pen, and brush are ignored.

### Implementation Notes

XVT/Mac  
 The icon must have a resource type of `ICON` or `CICN`.

XVT/Win  
 The icon must be declared in an `ICON` statement in the resource script.

XVT/PM  
 The icon must be declared in a `BITMAP` statement in the resource script.

XVT/XM  
 There must be an `ICON` definition in your resource manager file.

XVT/CH  
 This function isn't very useful because it merely displays the `rid` argument.

### Equivalent C Function

```
win_draw_icon()
```

---

## XVT\_BaseDrawProto::DrawLine

DRAW A LINE

---

### Prototypes

```
void  
DrawLine(  
    XVT_Pnt                pnt )
```

### Parameters

pnt  
The point to draw to.

### Description

This function draws a line from the current pen position to pnt.  
The pen position for subsequent drawing functions becomes pnt.

### Implementation Notes

XVT/CH  
Only horizontal and vertical lines are drawn accurately.

### Equivalent C Function

win\_draw\_line()

---

## XVT\_BaseDrawProto::DrawOval

DRAW AN OVAL

---

### Prototypes

```
void  
DrawOval(  
    XVT_Rct                boundary )
```

### Parameters

boundary  
The bounding rectangle for the oval. The bounding rectangle should not be empty.

### Description

This function draws an oval (ellipse) that is bounded by the rectangle boundary.

## Implementation Notes

XVT/CH

The rectangle given by boundary is drawn instead of an arc.

## Equivalent C Function

win\_draw\_oval()

---

# XVT\_BaseDrawProto::DrawPicture

DRAW A PICTURE

---

## Prototypes

```
void
DrawPicture(
    XVT_Rct          boundary,
    XVT_Picture*     pict )
```

## Parameters

boundary

The rectangle that bounds the drawn picture. The picture is scaled to fit the rectangle. For best results, the aspect ratio of boundary should be the same as the aspect ratio of the frame in which the picture was originally drawn.

pict

A pointer to the picture to draw.

## Description

Draws a picture in the window's client area.

## Implementation Notes

XVT/Mac

Pictures are Mac PICTs, which scale and stretch nicely.

XVT/Win, XVT/PM, XVT/XOL

Pictures are bitmaps, which tend to look “jaggy” when scaled or stretched. On these systems, drawing the picture in its original size is significantly faster than scaling or stretching it.

XVT/CH

Pictures are character maps. Scaling is ignored and the picture is simply clipped to boundary. To avoid any of these artifacts, you should draw the picture in its original size.

## Equivalent C Function

```
win_picture_draw()
```

---

# XVT\_BaseDrawProto::DrawPie

DRAW A PIE SECTION

---

## Prototypes

```
void  
DrawPie(  
    XVT_Rct          boundary,  
    XVT_Pnt          start  
    XVT_Pnt          stop )
```

## Parameters

**boundary**  
The bounding rectangle for the oval from which the pie is taken.  
The bounding rectangle should not be empty.

**start**  
Start vector.

**stop**  
Stop vector.

## Description

This function draws a section of an oval (a pie slice) in the client area of the window. The oval is bounded by **boundary**. An arc is drawn counter-clockwise along the oval, from the intersection of the start vector and the oval to the intersection of the stop vector and the oval. The pie is completed by drawing lines from the start and stop points to the center of the rectangle.

## Implementation Notes

XVT/CH  
The rectangle given by **boundary** is drawn instead of a pie slice.

## Equivalent C Function

```
win_draw_pie()
```

---

## XVT\_BaseDrawProto::DrawPolygon

DRAW A POLYGON

---

### Prototypes

```
void  
DrawPolygon(  
    XVT_Pnt*      pnts,  
    long          num )
```

### Parameters

**pnts**  
A pointer to an array of points that describe the vertices of a polygon.

**num**  
The number of points in pnts.

### Description

This function draws a polygon described by num vertices in the array pnts into the window's client area. If the starting and ending points don't coincide, an additional side is drawn to close the shape by connecting the starting and ending points, so there is an enclosed interior. The points are connected in the order found in the array. If any sides intersect, the determination of what's inside and what's outside is undefined.

For best performance, set the first point equal to the last point. Otherwise, XVT++ may have to allocate an array with num + 1 points in it and copy the original array to it.

### Implementation Notes

XVT/CH  
The polygon is rendered as though it were a polyline. No interior fill is done.

### Equivalent C Function

```
win_draw_polygon()
```

---

## XVT\_BaseDrawProto::DrawPolyline

DRAW A POLYLINE

---

### Prototypes

```
void
DrawPolyline(
    XVT_Pnt*      pnts,
    long          num )
```

### Parameters

**pnts**  
A pointer to an array of points that describe the vertices of a polygon.

**num**  
The number of points in pnts.

### Description

This function connects the num points in the pnts array with straight lines drawn in the window's client area. The last point is not automatically connected to the first; if you want a closed shape, make them the same. However, even if you create a closed shape, the shape is not considered to have an interior. If you want an interior, use DrawPolygon.

### Implementation Notes

XVT/CH  
Only horizontal and vertical lines are drawn accurately. Vertices between horizontal and vertical line segments are rendered with corner characters if available.

### Equivalent C Function

win\_draw\_polyline()

---

## XVT\_BaseDrawProto::DrawRect

DRAW A RECTANGLE

---

### Prototypes

```
void
DrawRect(
    XVT_Rct      boundary )
```

**Parameters**

boundary

The rectangle to be drawn. The rectangle should not be empty.

**Description**

Draws a rectangle in the window's client area.

A special usage of DrawRect is supported for inverting text to show a selection. To do that, use a hollow pen, a color of black, a solid brush, and a drawing mode of M\_XOR. Other combinations (e.g., a black pen) may display gaps between selection rectangles that are supposed to touch. The above combination doesn't have this problem.

**Equivalent C Function**

win\_draw\_rect()

---

## XVT\_BaseDrawProto::DrawRoundedRect

---

DRAW A RECTANGLE WITH ROUNDED CORNERS

---

**Prototypes**

```
void
DrawRoundedRect(
    XVT_Rct,          boundary
    long,             oval_width
    long              oval_height )
```

**Parameters**

boundary

The rectangle to be drawn. The rectangle should not be empty.

oval\_width

The width of the oval used for rounding corners.

oval\_height

The height of the oval used for rounding corners.

**Description**

This function draws a rectangle with rounded corners in the window's client area. Each corner is a quadrant of an oval that is oval\_width wide and oval\_height high.



**Implementation Notes**

XVT/CH

The rectangle does not have rounded corners.

**Equivalent C Function**`win_draw_round_rect()`

---

**XVT\_BaseDrawProto::DrawText**DRAW A TEXT STRING

---

**Prototypes**

```
void
DrawText(
    XVT_Pnt          pnt,
    const char*      str,
    long             len )
```

**Parameters**

**pnt**  
The point relative to which the text will be drawn. The text's baseline is at the point's y coordinate, and the left side of the first character starts at the point's x coordinate.

**str**  
The string to draw.

**len**  
The number of characters to draw. If `len` is `-1` the string is assumed to be null-terminated and the entire string is drawn.

**Description**

This function outputs the text string `str` starting at the point `pnt`, in the window's client area. The drawing is performed such that the text's baseline is at the point's y coordinate, and the left side of the first character starts at the point's x coordinate. For a diagram that depicts the positioning of text, see the "Drawing" chapter in the *XVT Guide*.

Text is drawn in the current font. The current pen and brush are ignored. Text is always drawn in the current foreground color.

Normally, only the "ink" making up the characters is transferred during drawing. Therefore, if text is drawn on top of existing graphics, the graphics will show through and around the text. However, if the current tools have been set to be opaque with

XVT\_DrawTools::SetOpaqueText( TRUE ), the text background is drawn in the current background color and existing graphics will not show through.

No ASCII control characters (e.g., tab, backspace, return) in the string are honored. Text layout implied by these controls must instead be achieved by drawing the text in segments and positioning each segment in the window appropriately. The appearance of strings containing such characters is undefined.

### Equivalent C Function

win\_draw\_text()

---

## XVT\_BaseDrawProto::GetBrush

RETRIEVE THE CURRENT BRUSH

---

### Prototypes

XVT\_Brush  
GetBrush() const

### Return Value

The window's current brush.

### Equivalent C Function

win\_get\_draw\_ctools()

---

## XVT\_BaseDrawProto::GetClip

RETRIEVE THE CURRENT CLIPPING RECTANGLE

---

### Prototypes

XVT\_Rct  
GetClip() const

### Return Value

The current clipping rectangle.

### Equivalent C Function

get\_clip()

---

## XVT\_BaseDrawProto::GetClipState

DETERMINE WHETHER CLIPPING IS ON OR OFF

---

### Prototypes

BOOLEAN  
GetClipState() const

### Return Value

A flag that is TRUE if clipping is enabled, FALSE if it is disabled.

---

## XVT\_BaseDrawProto::GetCurrentPoint

RETRIEVE THE CURRENT PEN POSITION

---

### Prototypes

XVT\_Pnt  
GetCurrentPoint() const

### Return Value

The current pen position.

---

## XVT\_BaseDrawProto::GetDrawMode

RETRIEVE THE CURRENT DRAWING MODE

---

### Prototypes

DRAW\_MODE  
GetDrawMode() const

### Return Value

The current drawing mode.

### Equivalent C Function

win\_get\_draw\_ctools()

---

## XVT\_BaseDrawProto::GetDrawTools

RETRIEVE THE CURRENT DRAWING TOOLS

---

### Prototypes

```
XVT_DrawTools  
GetDrawTools() const
```

### Return Value

The window's current drawing tools.

### Equivalent C Function

```
win_get_draw_ctools()
```

---

## XVT\_BaseDrawProto::GetFontMetrics

RETRIEVE A FONT'S LEADING, ASCENT AND DESCENT

---

### Prototypes

```
XVT_FontMetrics  
GetFontMetrics() const
```

### Return Value

The font metrics of the current font.

### Equivalent C Function

```
win_get_font_metrics()
```

---

## XVT\_BaseDrawProto::GetPen

RETRIEVE THE CURRENT PEN

---

### Prototypes

```
XVT_Pen  
GetPen() const
```

### Return Value

The window's current pen.

**Equivalent C Function**`win_get_draw_ctools()`

---

**XVT\_BaseDrawProto::GetTextWidth**

---

DETERMINE THE WIDTH OF A TEXT STRING

---

**Prototypes**

```
long
GetTextWidth(
    const char*    str,
    long           len ) const
```

**Parameters**

`str`  
A text string.

`len`  
The number of characters in the string. If `len` is `-1` the string is assumed to be null-terminated and the entire string is used.

**Return Value**

The width of the given string when set in the current font.

**Description**

This function gets the width in pixels of the text string `str` using the current font. This function is useful for calculating text layout, especially word wrapping.

To get the width of a string made of several different fonts (e.g., when the size or style varies), call `GetTextWidth` for the substrings that share a common font, then add up the widths. Using a `len` argument other than `-1` is handy for this because the substrings need not be null-terminated.

**Equivalent C Function**`win_get_text_width()`

---

## XVT\_BaseDrawProto::NeedsUpdate

DETERMINE IF AN AREA OF A WINDOW NEEDS TO BE DRAWN

---

### Prototypes

```
BOOLEAN  
NeedsUpdate(  
    XVT_Rct                boundary )
```

### Parameters

`boundary`  
The area to check for corruption.

### Return Value

A flag which is TRUE if any portion of the area bounded by `boundary` needs to be redrawn, FALSE if not.

### Description

When called from the context of an `e_update`, this function determines whether or not an area of the window needs to be redrawn. Note that the area delivered to `e_update` is the *extent* of all areas which need to be redrawn. It is not necessarily the case that the entire area needs to be redrawn.

### Equivalent C Function

```
needs_update()
```

---

## XVT\_BaseDrawProto::SetBackColor

SET THE CURRENT BACKGROUND COLOR

---

### Prototypes

```
void  
SetBackColor(  
    XVT_Color                color )
```

### Parameters

`color`  
The new background color.

## Description

Sets the window's background color. The background color is used for the spaces between hatch marks of a patterned brush, for the text background when text is opaque, and for the background of icons.

Do not confuse the background color set by this function with any sort of automatic background painting. Your application must explicitly paint a window in the background color during a call to `e_update`, usually by calling `Clear`.

## Equivalent C Function

`win_set_back_color()`

---

# XVT\_BaseDrawProto::SetBrush

SET THE CURRENT BRUSH

---

## Prototypes

```
void  
SetBrush(  
    XVT_Brush          brush )
```

## Parameters

`brush`  
The new brush.

## Description

Sets the window's current brush. Brushes are used for filling the interior of drawing primitives.

## Equivalent C Function

`win_set_cbrush()`

---

# XVT\_BaseDrawProto::SetClip

SET A WINDOW'S CLIPPING REGION

---

## Prototypes

```
void  
SetClip(  
    XVT_Rct          region )
```

**Parameters**

region  
The new clipping region. The region should not be empty.

**Description**

Sets a window's clipping region. The clipping region is a rectangular area bounded by the window's client area. If clipping is on, no drawing affects the area outside the clipping rectangle.

**Equivalent C Function**

set\_clip()

---

## XVT\_BaseDrawProto::SetClipState

TURN CLIPPING ON OR OFF

---

**Prototypes**

```
void
SetClipState(
    BOOLEAN          state )
```

**Parameters**

state  
A flag that is TRUE if clipping is to be enabled, FALSE if it is to be disabled.

**Description**

Turns clipping on or off.

---

## XVT\_BaseDrawProto::SetCurrentPoint

SET THE CURRENT PEN POSITION

---

**Prototypes**

```
void
SetCurrentPoint(
    XVT_Pnt      pnt )
```

**Parameters**

pnt  
The new pen position.



**Description**

Sets the current pen position. The pen position provides the starting point for the DrawLine and DrawALine functions.

**Equivalent C Function**

win\_move\_to()

---

## XVT\_BaseDrawProto::SetDrawMode

SET THE CURRENT DRAWING MODE

---

**Prototypes**

```
void
SetDrawMode(
    DRAW_MODE          mode )
```

**Parameters**

mode  
The new drawing mode.

**Description**

Sets the window's current drawing mode.

Drawing modes are defined by the DRAW\_MODE enumeration, which has at least the following members:

**M\_COPY**

The normal drawing mode. The source pixels are copied to the screen, erasing any destination pixels underneath them.

**M\_XOR**

The source is XOR'd with the inverse (NOT) of the destination. This mode has the property that drawing the same thing twice is guaranteed to have no effect and that drawing something once will be visible under most combinations of foreground and background colors.

**M\_OR**

The source pixels are OR'd with the destination pixels and the result is displayed on the screen.

**M\_CLEAR**

If the source pixel is set, it is written to the screen. The destination pixels are ignored.

**M\_NOT\_COPY**

The inverse of the source pixels is copied to the screen.

**M\_NOT\_XOR**

The inverse (NOT) of the source is XOR'd with the inverse (NOT) of the destination.

**M\_NOT\_CLEAR**

If the source pixel is not set, its inverse is written to the screen. The destination pixels are ignored.

**Implementation Notes**

Use of modes other than M\_COPY for printing is not portable.

**Equivalent C Function**

`win_set_draw_mode()`

---

## XVT\_BaseDrawProto::SetDrawTools

SET THE CURRENT DRAWING TOOLS

---

**Prototypes**

```
void
SetDrawTools(
    XVT_DrawTools    tools )
```

**Parameters**

`tools`  
The new drawing tools.

**Description**

Sets the window's current drawing tools. The drawing tools control the attributes of all the drawing primitives.

**Equivalent C Function**

`win_set_draw_ctools()`

---

## XVT\_BaseDrawProto::SetFont

SET THE FONT USED FOR DRAWING TEXT

---

**Prototypes**

```
void
SetFont(
    XVT_Font        font )
```

**Parameters**

font

The font that will become the current font. It should have been received by an `e_font` call or through `GetDrawTools`.

**Description**

This function sets the font to be used for all subsequent calls to `DrawText`.

**Implementation Notes**

XVT/CH

The current font is ignored. All drawing is done in whatever font the screen supports.

**Equivalent C Function**

`win_set_font()`

---

## XVT\_BaseDrawProto::SetForeColor

SET THE CURRENT FOREGROUND COLOR

---

**Prototypes**

```
void  
SetForeColor(  
    XVT_Color      color )
```

**Parameters**

color

The new foreground color.

**Description**

Sets the window's foreground color.

The foreground color is used only for drawing text and icons. Other drawing primitives take their colors from the current pen and brush.

**Equivalent C Function**

`win_set_fore_color()`

---

## XVT\_BaseDrawProto::SetPen

SET THE CURRENT PEN

---

### Prototypes

```
void  
SetPen(  
    XVT_Pen  
    pen )
```

### Parameters

pen  
The new pen.

### Description

Sets the window's current pen. Pens are used for drawing the outlines of drawing primitives.

### Equivalent C Function

win\_set\_cpen()

---

## XVT\_BaseDrawProto::UpdateWindow

CAUSE A WINDOW'S APPEARANCE TO BE MADE CURRENT

---

### Prototypes

```
void  
UpdateWindow()
```

### Description

Forces all pending calls to `e_update` to be made. If calls are pending they will be made recursively, before `UpdateWindow` returns.

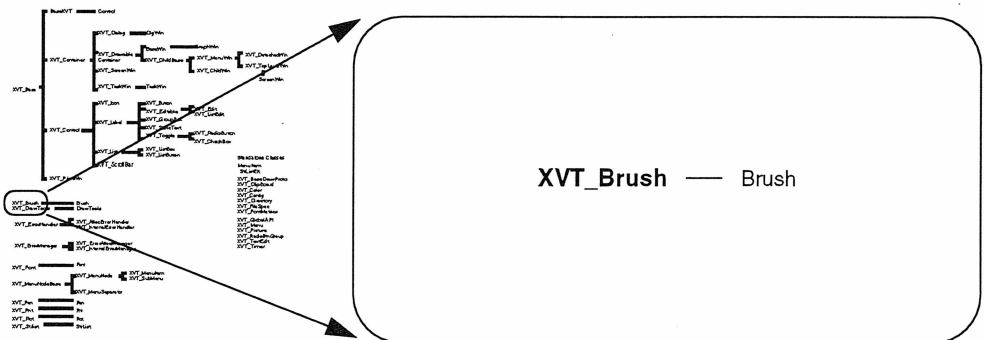
### Equivalent C Function

update\_window()

## Implementation Members

```
XVT_BaseDrawProto  
~XVT_BaseDrawProto  
CurrentPoint  
ClipState
```

# XVT\_Brush



# Overview

<b>Header File</b>	tools.h
<b>Source File</b>	tools.cc
<b>Superclass</b>	
<b>Subclasses</b>	Brush
<b>Usage</b>	Concrete

Instances of the brush class describe how interior regions of drawing primitives can be filled.

For brushes with patterns other than `PAT_HOLLOW` or `PAT_SOLID`, the interior area is first filled with the background color from the window's current `XVT_DrawTools` instance; then the hatching is drawn in the brush's color.

## Constructors

```
XVT_Brush()  
XVT_Brush( PAT_STYLE pattern, XVT_Color color )  
    Create a new brush with the given pattern and color. Equivalent  
    to using the default constructor, then SetPattern and SetColor.  
XVT_Brush( const XVT_Brush& brush )  
~XVT_Brush()
```

## Operators

```
XVT_Brush& operator=( const XVT_Brush& brush )  
BOOLEAN operator==( const XVT_Brush& brush )  
    Brushes can be assigned and compared for equality.
```

## Member Functions

---

### XVT\_Brush::GetColor

RETRIEVE THE BRUSH'S COLOR

---

#### Prototypes

```
XVT_Color  
GetColor() const
```

#### Return Value

The brush's current color.

---

### XVT\_Brush::GetPattern

RETRIEVE A BRUSH'S PATTERN

---

#### Prototypes

```
PAT_STYLE  
GetPattern() const
```

#### Return Value

The brush's pattern.

---

## XVT\_Brush::SetColor

SET THE BRUSH'S COLOR

---

### Prototypes

```
void  
SetColor(  
    XVT_Color          color )
```

### Parameters

color  
The brush's new color.

### Description

Sets the brush's color.

---

## XVT\_Brush::SetPattern

SET A BRUSH'S PATTERN

---

### Prototypes

```
void  
SetPattern(  
    PAT_STYLE          pattern )
```

### Parameters

pattern  
The new pattern.

### Description

The PAT\_STYLE enumeration defines the following patterns that are usable in brushes:

PAT\_HOLLOW  
No interior fill.

PAT\_SOLID  
Fill the interior with a solid color.

PAT\_HORZ  
Fill the interior with horizontal lines.

PAT\_VERT  
Fill the interior with vertical lines.

**PAT\_FDIAG**

Fill the interior with forward-leaning diagonal lines.

**PAT\_BDIAG**

Fill the interior with backward-leaning diagonal lines.

**PAT\_CROSS**

Fill the interior with a grid.

**PAT\_DIAGCROSS**

Fill the interior with a diagonal grid.

## **Implementation Members**

ConvertTo

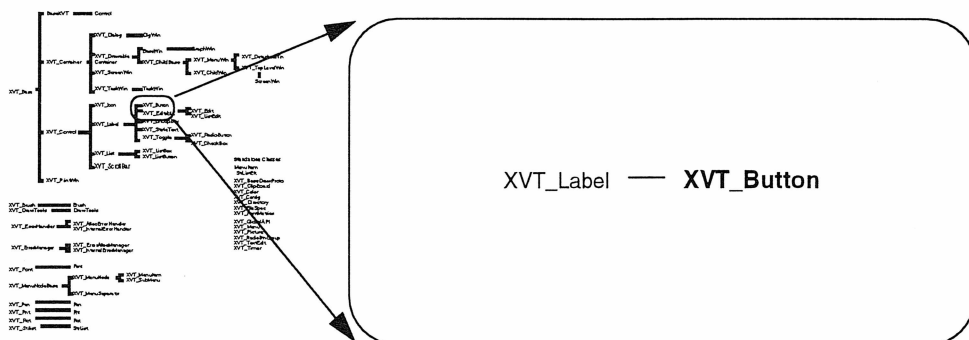
ConvertFrom

Pattern

Color



## XVT Button



## Overview

<b>Header File</b>	button.h
<b>Source File</b>	
<b>Superclass</b>	XVT_Label
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The `XVT_Button` class defines the interface to all buttons.

You use this class by creating a subclass that overrides virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

`XVT_Button( XVT_Dialog* parent, long cid )`  
Create a button in a dialog.

```
XVT_Button( XVT_DrawableContainer* parent, long cid )
```

Create a button in a window.

## Member Functions

## XVT\_Button::e\_action

RECEIVE NOTIFICATION THAT BUTTON HAS BEEN OPERATED

## Prototypes

```
virtual void
e_action()
```

### Description

This member function is called when a button has been operated. The default version does nothing. Your subclass should provide a definition for this function that does whatever you want to do when a button is pressed.

## Inherited Member Functions

### From XVT\_Label

```
page 239 void GetTitle( char* str, unsigned long* len )
```

```

page 239  virtual BOOLEAN Init( XVT_Rct boundary, long = 0L, char *
           = NULL )

```

page 240      void SetTitle( char\* str )

## From XVT\_Control

page 92      virtual void Close()

page 93      virtual void e\_create()

page 93      virtual void e\_destroy()

```

page 94      virtual long e_user( long id, void *data )

```

page 95      BOOLEAN GetEnabledState()

page 95      long GetID( void )

page 95      XVT\_Base \*GetParent( void )

page 96      BOOLEAN GetVisibleState()

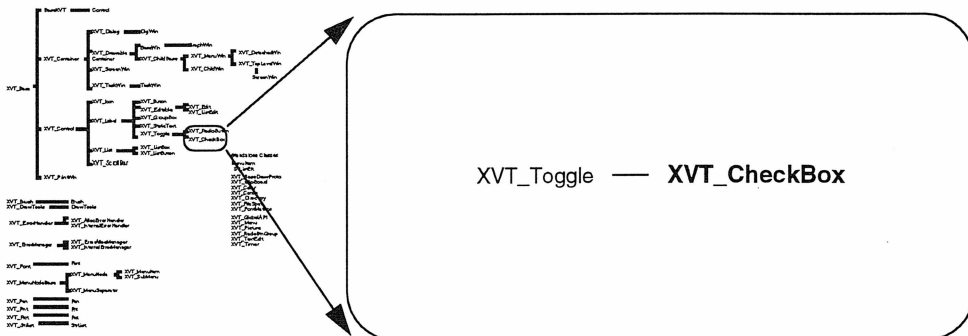
page 96      void Init()

*page 96*     void MakeFront()  
*page 97*     void SetEnabledState( BOOLEAN state )  
*page 98*     void SetInnerRect( XVT\_Rct boundary )  
*page 98*     void SetVisibleState( BOOLEAN state )

### From XVT\_Base

*page 11*     virtual BaseWin\* CastToBaseWin()  
*page 10*     virtual DlgWin\* CastToDlgWin()  
*page 10*     virtual ScreenWin\* CastToScreenWin11()  
*page 10*     virtual TaskWin\* CastToTaskWin11()  
*page 11*     virtual XVT\_Button \*CastToButton()  
*page 11*     virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*     virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*     virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*     virtual XVT\_Dialog \*CastToDialog()  
*page 11*     virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*     virtual XVT\_Edit \*CastToEdit()  
*page 11*     virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*     virtual XVT\_Icon \*CastToIcon()  
*page 11*     virtual XVT\_ListBox \*CastToListBox()  
*page 11*     virtual XVT\_ListButton \*CastToListButton()  
*page 11*     virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*     virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*     virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*     virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*     virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*     virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*     virtual XVT\_StaticText \*CastToStaticText()  
*page 11*     virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*     virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*     virtual XVT\_Rct GetInnerRect()  
*page 13*     virtual XVT\_Rct GetOuterRect()

# XVT\_CheckBox



## Overview

<b>Header File</b>	checkbox.h
<b>Source File</b>	checkbox.cc
<b>Superclass</b>	XVT_Toggle
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The XVT\_CheckBox class defines the interface to all check boxes.

You use this class by creating a subclass that overrides virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

XVT\_CheckBox( XVT\_Dialog\* parent, long cid )  
Create a check box in a dialog.

XVT\_CheckBox( XVT\_DrawableContainer\* parent, long cid )  
Create a check box in a window.

virtual ~XVT\_CheckBox()

## Member Functions

---

### XVT\_CheckBox::SetCheckedState

---

CHECK OR UNCHECK A CHECK BOX

---

#### Prototypes

```
void
SetCheckedState(
    BOOLEAN          state )
```

#### Parameters

*state*  
A flag which is TRUE if the checkbox is to be checked and FALSE if it is to be unchecked.

#### Description

Check or uncheck a checkbox.

#### Equivalent C Function

`win_check_box()`

## Inherited Member Functions

#### From XVT\_Toggle

*page 394*    virtual void e\_action()

*page 394*    virtual BOOLEAN GetCheckedState()

#### From XVT\_Label

*page 239*    void GetTitle( char\* str, unsigned long\* len )

*page 239*    virtual BOOLEAN Init( XVT\_Rct boundary, long = 0L, char \*  
= NULL )

*page 240*    void SetTitle( char\* str )

**From XVT\_Control**

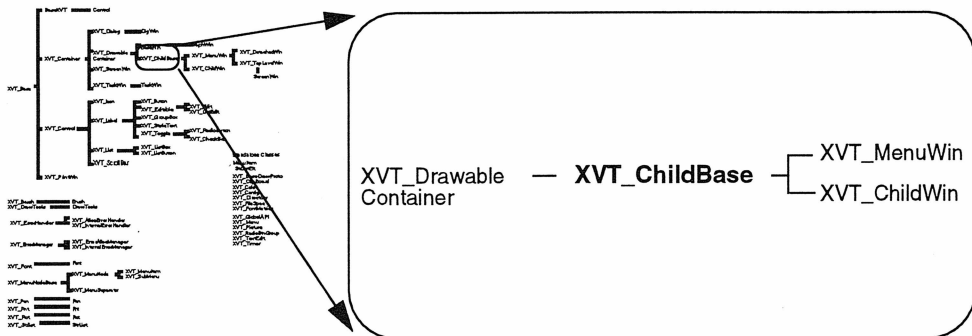
*page 92*    virtual void Close()  
*page 93*    virtual void e\_create()  
*page 93*    virtual void e\_destroy()  
*page 94*    virtual long e\_user( long id, void \*data )  
*page 95*    BOOLEAN GetEnabledState()  
*page 95*    long GetID( void )  
*page 95*    XVT\_Base \*GetParent( void )  
*page 96*    BOOLEAN GetVisibleState()  
*page 96*    void Init()  
*page 96*    void MakeFront()  
*page 97*    void SetEnabledState( BOOLEAN state )  
*page 98*    void SetInnerRect( XVT\_Rct boundary )  
*page 98*    void SetVisibleState( BOOLEAN state )

**From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()

<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>

# XVT\_ChildBase



## Overview

<b>Header File</b>	childb.h
<b>Source File</b>	childb.cc
<b>Superclass</b>	XVT_DrawableContainer
<b>Subclasses</b>	XVT_ChildWin, XVT_MenuWin
<b>Usage</b>	Implementation

The ChildBase class defines the interface common to all windows that may have children.



## Member Functions

---

### XVT\_ChildBase::e\_hscroll

RECEIVE NOTIFICATION OF ACTIVITY ON A WINDOW'S HORIZONTAL SCROLLBAR

---

#### Prototypes

```
virtual void  
e_hscroll(  
    SCROLL_CONTROL    activity,  
    long               pos )
```

#### Parameters

**activity**  
The site of the scrollbar activity.

**pos**  
The new thumb position.

#### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to activity on a window's horizontal scrollbar.

This function is identical in behavior to the XVT\_ScrollBar::e\_action member function.

---

### XVT\_ChildBase::e\_vscroll

RECEIVE NOTIFICATION OF ACTIVITY ON A WINDOW'S VERTICAL SCROLLBAR

---

#### Prototypes

```
virtual void  
e_vscroll(  
    SCROLL_CONTROL    activity,  
    long               pos )
```

#### Parameters

**activity**  
The site of the scrollbar activity.

**pos**  
The new thumb position.

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to activity on a window's vertical scrollbar.

This function is identical in behavior to the `XVT_ScrollBar::e_action` member function.

---

**XVT\_ChildBase::GetActiveTextEdit**

RETRIEVE THE CURRENTLY ACTIVE TEXT EDIT

---

**Prototype**

```
XVT_TextEdit*
GetActiveTextEdit()
```

**Return Value**

A pointer to the currently active text edit object or `NULL` if no text edit object is active.

**Description**

Retrieve a pointer to the currently active text edit object.

**Equivalent C Function**

`tx_get_active`

---

**XVT\_ChildBase::GetCaretPos**

RETRIEVE THE WINDOW'S CURRENT CARET POSITION

---

**Prototypes**

```
XVT_Pnt
GetCaretPos() const
```

**Return Value**

The window's current caret position.

---

## XVT\_ChildBase::GetCaretState

DETERMINE IF A WINDOW'S CARET IS VISIBLE OR INVISIBLE

---

### Prototypes

```
BOOLEAN  
GetCaretState() const
```

### Return Value

A flag that is TRUE if the caret is visible, FALSE if invisible.

---

## XVT\_ChildBase::GetEnabledState

DETERMINE WHETHER A WINDOW IS ENABLED OR DISABLED

---

### Prototypes

```
BOOLEAN  
GetEnabledState() const
```

### Return Value

A flag that is TRUE if the window is enabled, FALSE if not.

---

## XVT\_ChildBase::GetParent

RETRIEVE THE PARENT WINDOW

---

### Prototypes

```
virtual XVT_Base*  
GetParent() const
```

### Return Value

A pointer to the parent window.

### Description

Usually, the parent window was given as a parameter when the window was created. The parent of any top-level window or dialog is always the task window. The parent of the task window or any detached window is the screen window.

**Equivalent C Function**`get_parent()`

---

**XVT\_ChildBase::GetScrollPosition**RETRIEVE A BORDER SCROLLBAR'S CURRENT POSITION

---

**Prototypes**

```
long  
GetScrollPosition(  
    SCROLL_TYPE          scroll_type ) const
```

**Parameters**

```
scroll_type  
    Which border scrollbar to operate on. Valid values are:  
    HSCROLL  
        Operate on the horizontal border scrollbar.  
    VSCROLL  
        Operate on the vertical border scrollbar.
```

**Return Value**

The border scrollbar's current thumb position.

**Equivalent C Function**`get_scroll_pos()`

---

**XVT\_ChildBase::GetScrollProportion**RETRIEVE A BORDER SCROLLBAR'S THUMB PROPORTION

---

**Prototypes**

```
long  
GetScrollProportion(  
    SCROLL_TYPE          scroll_type ) const
```

**Parameters**`scroll_type`

Which border scrollbar to operate on. Valid values are:

`HSCROLL`

Operate on the horizontal border scrollbar.

`VSCROLL`

Operate on the vertical border scrollbar.

**Return Value**

The border scrollbar's current thumb proportion.

**Equivalent C Function**`get_scroll_proportion()`

---

**XVT\_ChildBase::GetScrollRange**RETRIEVE A BORDER SCROLLBAR'S RANGE

---

**Prototypes**

```
void
GetScrollRange(
    SCROLL_TYPE      scroll_type,
    long*             min,
    long*             max ) const
```

**Parameters**`scroll_type`

Which border scrollbar to operate on. Valid values are:

`HSCROLL`

Operate on the horizontal border scrollbar.

`VSCROLL`

Operate on the vertical border scrollbar.

`min`

A pointer to storage to receive the minimum value of the scrollbar range.

`max`

A pointer to storage to receive the maximum value of the scrollbar range.

**Description**

Retrieves a border scrollbar's range.

**Equivalent C Function**`get_scroll_range()`

---

**XVT\_ChildBase::GetTextEdit**RETRIEVE A TEXT EDIT OBJECT BASED ON ID

---

**Prototype**

```
XVT_TextEdit*
GetTextEdit(
    long id )
```

**Parameters**

id  
The text edit's control ID.

**Return Value**

A pointer to the corresponding text edit object or NULL if no text edit object with a matching ID was found.

**Description**

This function is used to retrieve text edit objects created from resources. When you create a text edit dynamically, there is no need to use this function because the new operator gives you a pointer to it.

**Equivalent C Function**`get_tx_edit`

---

**XVT\_ChildBase::GetVisibleState**DETERMINE IF A WINDOW IS VISIBLE

---

**Prototypes**

```
BOOLEAN
GetVisibleState() const
```

**Return Value**

A flag that is TRUE if the window is visible, FALSE if not.

---

## XVT\_ChildBase::MakeFront

MAKE A WINDOW BE FRONTMOST AND GIVE IT KEYBOARD FOCUS

---

### Prototypes

```
void  
MakeFront()
```

### Description

Makes a window be frontmost in the occlusion order and gives it the keyboard focus.

### Implementation Notes

XVT/Win

It is not possible to make a window appear in front of any type of dialog, modal or modeless.

XVT/Mac

It is not possible to make a window appear in front of a modal dialog.

### Equivalent C Function

```
set_front_window()
```

---

## XVT\_ChildBase::ReleaseMouse

RELEASE A PREVIOUSLY TRAPPED MOUSE

---

### Prototypes

```
void  
ReleaseMouse()
```

### Description

Releases a previously trapped mouse.

### Equivalent C Function

```
release_mouse()
```

---

## XVT\_ChildBase::SetCaretDimensions

SET THE DIMENSIONS OF A WINDOW'S CARET

---

### Prototypes

```
void  
SetCaretDimensions(  
    XVT_Pnt                vector )
```

### Parameters

vector  
The caret's dimension vector (height and width). If vector.X is zero the default native caret width is used. If vector.Y is zero then the caret height changes dynamically according to the height of the current font.

### Description

Sets the dimensions of a window's caret.

If you never call this function, the caret assumes a height appropriate for the current font. Therefore, if you only display one font in a window, calling SetCaretDimensions is superfluous.

### Implementation Notes

XVT/CH  
Caret dimensions are ignored.

### Equivalent C Function

set\_caret\_dimensions()

---

## XVT\_ChildBase::SetCaretPos

SET THE WINDOW'S CARET POSITION

---

### Prototypes

```
void  
SetCaretPos(  
    XVT_Pnt                point )
```

### Parameters

point  
The new caret position.



**Description**

Sets the window's caret position.

**Equivalent C Function**

caret\_on()

---

## XVT\_ChildBase::SetCaretState

MAKE A WINDOW'S CARET VISIBLE OR INVISIBLE

---

**Prototypes**

```
void  
SetCaretState(  
    BOOLEAN                state )
```

**Parameters**

state  
A flag that is TRUE if the caret is to be visible, FALSE if invisible.

**Description**

Makes a window's caret visible or invisible. The caret is usually used to indicate the text insertion point—where characters typed at the keyboard will appear.

**Equivalent C Function**

caret\_off()  
caret\_on()

---

## XVT\_ChildBase::SetCursor

SET A WINDOW'S CURSOR SHAPE

---

**Prototypes**

```
void  
SetCursor(  
    CURSOR                cursor )
```

**Parameters**

`cursor`  
 The new cursor.  
 Valid types of cursors are:  
`CURSOR_ARROW`  
 The standard system arrow.  
`CURSOR_IBEAM`  
 An I-Beam-style cursor typically used for selecting text.  
`CURSOR_CROSS`  
 A cross-hair cursor.  
`CURSOR_PLUS`  
 A plus sign.  
`CURSOR_WAIT`  
 The standard wait cursor.  
`CURSOR_USER + N`  
 A user-defined cursor. N starts at zero.

**Description**

The cursor is the shape that indicates the current mouse position. When the cursor is inside a window's client area or when the mouse is trapped to a window, the cursor is rendered using the window's cursor instead of the standard system pointer.

**Implementation Notes**

XVT/CH  
 There is only one cursor, a blinking block. This function is ignored. Your application should not rely on the cursor shape to convey information.

**Equivalent C Function**

`set_cursor()`

---

## XVT\_ChildBase::SetEnabledState

ENABLE OR DISABLE A WINDOW

---

**Prototypes**

```
void
SetEnabledState(
    BOOLEAN          state )
```

**Parameters**

state

A flag that is TRUE if the window is to be enabled, FALSE if it is to be disabled.

**Description**

Enables or disables a window according to the state parameter. When a window is disabled, its e\_focus, e\_mouse\_\* and e\_char event handler member functions are not called and those events are directed to the window's parent.

**Equivalent C Function**

enable\_window()

---

## XVT\_ChildBase::SetScrollPosition

---

SET THE THUMB POSITION OF A BORDER SCROLLBAR

---

**Prototypes**

```
void
SetScrollPosition(
    SCROLL_TYPE      scroll_type,
    long              position )
```

**Parameters**

scroll\_type

Which border scrollbar to operate on. Valid values are:

HSCROLL

Operate on the horizontal border scrollbar.

VSCROLL

Operate on the vertical border scrollbar.

position

The border scrollbar's new thumb position. It must be the case that SHRT\_MIN < position < SHRT\_MAX.

**Description**

Sets the thumb position of a window's border scrollbar.

**Equivalent C Function**

set\_scroll\_pos()

---

## XVT\_ChildBase::SetScrollProportion

SET A BORDER SCROLLBAR'S THUMB PROPORTION

---

### Prototypes

```
void  
SetScrollProportion(  
    SCROLL_TYPE      scroll_type,  
    long              proportion )
```

### Parameters

**scroll\_type**  
Which border scrollbar to operate on. Valid values are:  
HSCROLL  
    Operate on the horizontal border scrollbar.  
VSCROLL  
    Operate on the vertical border scrollbar.

**proportion**  
The scrollbar's new proportion.

### Description

Sets a border scrollbar's thumb proportion.

### Equivalent C Function

```
set_scroll_proportion()
```

---

## XVT\_ChildBase::SetScrollRange

SET A BORDER SCROLLBAR'S RANGE

---

### Prototypes

```
void  
SetScrollRange(  
    SCROLL_TYPE      scroll_type,  
    long              min,  
    long              max,  
    long              pos )
```

**Parameters****scroll\_type**

Which border scrollbar to operate on. Valid values are:

HSCROLL

Operate on the horizontal border scrollbar.

VSCROLL

Operate on the vertical border scrollbar.

**min**The minimum value of the scrollbar range. It must be the case that `SHRT_MIN < min < SHRT_MAX`.**max**The maximum value of the scrollbar range. It must be the case that `SHRT_MIN < max < SHRT_MAX`.**pos**

The scrollbar thumb position in the new range.

**Description**

Sets a border scrollbar's range.

**Equivalent C Function**`set_scroll_range()`

---

**XVT\_ChildBase::SetVisibleState**MAKE A WINDOW VISIBLE OR INVISIBLE

---

**Prototypes**

```
void
SetVisibleState(
    BOOLEAN                state )
```

**Parameters****state**

A flag that is TRUE if the window is to be visible, FALSE if it is to be invisible.

**Description**

This function makes a window visible or invisible. An invisible window does not appear on the screen and cannot have focus or receive input events. If a window with focus is made invisible, focus is transferred to another window within the application or to the task window if there are no other top level windows. Since the window

cannot receive input events, the event handler member functions `e_char` and `e_mouse_*` are not called.

### Equivalent C Function

`show_window()`

---

## XVT\_ChildBase::TrapMouse

TRAP THE MOUSE

---

### Prototypes

```
void
TrapMouse()
```

### Description

Traps the mouse to this window. When the mouse is trapped, *all* mouse events are sent to this window, even if the mouse is outside of the window. This means that the mouse cannot be used to operate anything outside of the window's client area.

Note that the mouse coordinates in mouse events may lie outside the window's client area. Your implementations of the `e_mouse_*` event handlers should take this into account, perhaps by using `XVT_Rct::Constrain` to force mouse coordinates to lie within the client area.

Calls to the trapping window's `e_mouse_move` handler are generated continuously as long as the mouse is trapped.

The effects of trapping the mouse more than once are *undefined*.

### Equivalent C Function

`trap_mouse()`

## Implementation Members

```
XVT_ChildBase
~XVT_ChildBase
Parent
EnableProtocol
ShowProtocol
ScrollProtocol
HasVertScroll
```

HasHorzScroll  
EnabledState  
VisibleState  
CaretState  
CaretPos  
CreateTxList  
CreateTextEdits

## Inherited Member Functions

### From XVT\_DrawableContainer

*page 129*    void Clear()  
*page 129*    void Clear( XVT\_Color color )  
*page 129*    void Close()  
*page 128*    XVT\_BaseDrawProto\* DrawProtocol  
*page 130*    virtual void e\_char(  
            short chr,  
            BOOLEAN shift,  
            BOOLEAN control)  
*page 131*    virtual void e\_create()  
*page 132*    virtual void e\_destroy()  
*page 132*    virtual void e\_focus( BOOLEAN active )  
*page 133*    virtual void e\_mouse\_dbl(  
            XVT\_Pntpoint,  
            BOOLEAN shift,  
            BOOLEAN control,  
            short button )  
*page 134*    virtual void e\_mouse\_down(  
            XVT\_Pntpoint,  
            BOOLEAN shift,  
            BOOLEAN control,  
            short button )  
*page 135*    virtual void e\_mouse\_move(  
            XVT\_Pntpoint,  
            BOOLEAN shift,  
            BOOLEAN control,  
            short button )

<i>page 135</i>	virtual void e_mouse_up( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 136</i>	virtual void e_size( XVT_Rct boundary )
<i>page 137</i>	virtual void e_timer( long id )
<i>page 137</i>	virtual void e_update( XVT_Rct boundary )
<i>page 139</i>	virtual long e_user( long id, void *data )
<i>page 140</i>	XVT_Control *GetCtl( long cid )
<i>page 140</i>	long GetCtlCount()
<i>page 141</i>	EVENT_MASK GetEventMask() const
<i>page 141</i>	XVT_Control *GetFirstCtl()
<i>page 142</i>	XVT_ChildBase *GetFirstWin()
<i>page 142</i>	XVT_Control *GetNextCtl()
<i>page 143</i>	XVT_ChildBase *GetNextWin()
<i>page 143</i>	long GetWinCount()
<i>page 144</i>	void Invalidate()
<i>page 144</i>	void Invalidate( XVT_Rctregion )
<i>page 145</i>	void Scroll( XVT_Rct boundary, long dh, long dv )
<i>page 146</i>	void SetEventMask( EVENT_MASK ask )
<i>page 148</i>	void SetInnerRect( XVT_Rct r )

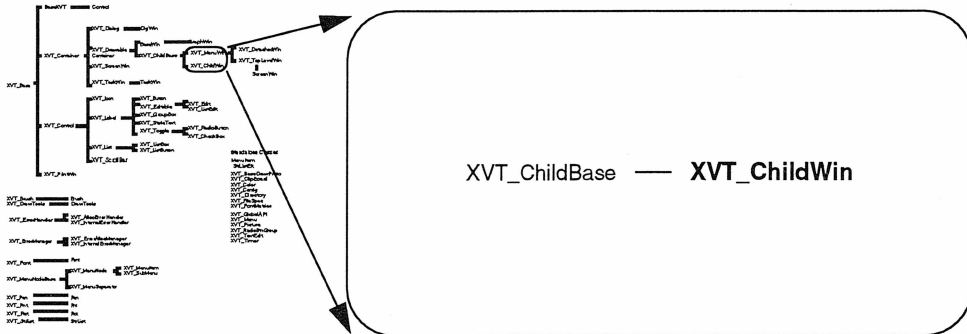
### From XVT\_Base

<i>page 11</i>	virtual BaseWin* CastToBaseWin()
<i>page 10</i>	virtual DlgWin* CastToDlgWin()
<i>page 10</i>	virtual ScreenWin* CastToScreenWin11()
<i>page 10</i>	virtual TaskWin* CastToTaskWin11()
<i>page 11</i>	virtual XVT_Button *CastToButton()
<i>page 11</i>	virtual XVT_CheckBox *CastToCheckBox()



*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_ChildWin



# Overview

<b>Header File</b>	child.h
<b>Source File</b>	child.cc
<b>Superclass</b>	XVT_ChildBase
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The `XVT_ChildWin` class defines the interface to all child windows.

This class is an abstract GUI object class. You can instantiate it but the instances will not respond to events.

You use this class by creating a subclass that overrides virtual event handling member functions with implementations that actually do something in response to events.

# Constructors

```
XVT_ChildWin( XVT_ChildBase* parent )
    Create a child window in the given parent.
~XVT_ChildWin()
```

## Member Functions

---

### XVT\_ChildWin::Init

INITIALIZE A CHILD WINDOW

---

#### Prototypes

```

BOOLEAN
Init(
    WIN_TYPE
    XVT_Rct
    long
    wtype,
    boundary,
    flags )

BOOLEAN
Init(
    long
    rid )

```

#### Parameters

**wtype**  
The type of window to be created. It should be either `W_PLAIN` or `W_NO_BORDER`.

**boundary**  
The bounding rectangle (in pixels) of the window's client area. The rectangle is relative to the parent window's client area.

**flags**  
A bitwise OR'd combination of flags that control the window's attributes and decoration.

**rid**  
The resource ID by means of which the window's dimensions, attributes, and contents can be located.

#### Return Value

TRUE if the window was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by windows. Recovery may be attempted by disposing of the new window, closing another window, and retrying the creation of the window.

#### Description

The `Init` member functions create the native window and call the window's `se_create` method. When execution returns from the `Init` call, the window is complete and ready to use. Prior to the `Init` call the window is not usable.

Init( wtype, boundary, flags )

Creates only a window with the given parameters. XVT++ control objects must be created separately by the user.

Init( rid )

Creates a window and contained controls from a resource specification. XVT++ control objects corresponding to the controls described in the resource must be created and installed separately by the application developer. The recommended place to do this is in the window's `e_create` member function; however, the control objects can be created at any time. Events intended for controls that have no corresponding XVT++ control object will cause a run-time error.

### Equivalent C Function

`create_window()`

`create_def_window()`

`create_res_window()`

## Implementation Members

`BOOLEAN Init( XVT_WindowDef* def )`

`GetMenuWinAncestor`

## Inherited Member Functions

### From XVT\_ChildBase

*page 49*    `virtual void e_hscroll( SCROLL_CONTROL activity, short pos )`

*page 49*    `virtual void e_vscroll( SCROLL_CONTROL activity, short pos )`

*page 50*    `XVT_TextEdit* GetActiveTextEdit()`

*page 50*    `XVT_Pnt GetCaretPos() const`

*page 51*    `BOOLEAN GetCaretState() const`

*page 51*    `BOOLEAN GetEnabledState()`

*page 51*    `XVT_ChildBase *GetParent() const`

*page 52*    `long GetScrollPosition( SCROLL_TYPE scroll_type ) const`

*page 52*    `long GetScrollProportion( SCROLL_TYPE scroll_type ) const`

<i>page 53</i>	<code>void GetScrollRange( SCROLL_TYPE scroll_type, long *min, long *max ) const</code>
<i>page 54</i>	<code>XVT_TextEdit* GetTextEdit( long id )</code>
<i>page 54</i>	<code>BOOLEAN GetVisibleState()</code>
<i>page 55</i>	<code>void MakeFront()</code>
<i>page 55</i>	<code>void ReleaseMouse()</code>
<i>page 56</i>	<code>void SetCaretDimensions( XVT_Pnt vector )</code>
<i>page 56</i>	<code>void SetCaretPos( XVT_Pnt point )</code>
<i>page 57</i>	<code>void SetCaretState( BOOLEAN state )</code>
<i>page 57</i>	<code>void SetCursor( CURSOR cursor )</code>
<i>page 58</i>	<code>void SetEnabledState( BOOLEAN state )</code>
<i>page 59</i>	<code>void SetScrollPosition( SCROLL_TYPE scroll_type, long position )</code>
<i>page 60</i>	<code>void SetScrollProportion( SCROLL_TYPE scroll_type, long proportion )</code>
<i>page 60</i>	<code>void SetScrollRange( SCROLL_TYPE scroll_type, long min, long max, long pos )</code>
<i>page 61</i>	<code>void SetVisibleState( BOOLEAN f )</code>
<i>page 62</i>	<code>void TrapMouse()</code>

### **From XVT\_DrawableContainer**

<i>page 129</i>	<code>void Clear()</code>
<i>page 129</i>	<code>void Clear( XVT_Color color )</code>
<i>page 129</i>	<code>void Close()</code>
<i>page 128</i>	<code>XVT_BaseDrawProto* DrawProtocol</code>
<i>page 130</i>	<code>virtual void e_char( short chr, BOOLEAN shift, BOOLEAN control)</code>
<i>page 131</i>	<code>virtual void e_create()</code>
<i>page 132</i>	<code>virtual void e_destroy()</code>
<i>page 132</i>	<code>virtual void e_focus( BOOLEAN active )</code>

<i>page 133</i>	virtual void e_mouse_dbl( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 134</i>	virtual void e_mouse_down( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_move( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_up( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 136</i>	virtual void e_size( XVT_Rct boundary )
<i>page 137</i>	virtual void e_timer( long id )
<i>page 137</i>	virtual void e_update( XVT_Rct boundary )
<i>page 139</i>	virtual long e_user( long id, void *data )
<i>page 140</i>	XVT_Control *GetCtl( long cid )
<i>page 140</i>	long GetCtlCount()
<i>page 141</i>	EVENT_MASK GetEventMask() const
<i>page 141</i>	XVT_Control *GetFirstCtl()
<i>page 142</i>	XVT_ChildBase *GetFirstWin()
<i>page 142</i>	XVT_Control *GetNextCtl()
<i>page 143</i>	XVT_ChildBase *GetNextWin()
<i>page 143</i>	long GetWinCount()
<i>page 144</i>	void Invalidate()
<i>page 144</i>	void Invalidate( XVT_Rctregion )
<i>page 145</i>	void Scroll( XVT_Rct boundary, long dh, long dv )

*page 146*     void SetEventMask( EVENT\_MASK ask )

*page 148*     void SetInnerRect( XVT\_Rct r )

### **From XVT\_Base**

*page 11*     virtual BaseWin\* CastToBaseWin()

*page 10*     virtual DlgWin\* CastToDlgWin()

*page 10*     virtual ScreenWin\* CastToScreenWin11()

*page 10*     virtual TaskWin\* CastToTaskWin11()

*page 11*     virtual XVT\_Button \*CastToButton()

*page 11*     virtual XVT\_CheckBox \*CastToCheckBox()

*page 11*     virtual XVT\_ChildWin \*CastToChildWin()

*page 11*     virtual XVT\_DetachedWin \*CastToDetachedWin()

*page 11*     virtual XVT\_Dialog \*CastToDialog()

*page 11*     virtual XVT\_DrawableContainer\*CastToDrawableContainer()

*page 11*     virtual XVT\_Edit \*CastToEdit()

*page 11*     virtual XVT\_GroupBox \*CastToGroupBox()

*page 11*     virtual XVT\_Icon \*CastToIcon()

*page 11*     virtual XVT\_ListBox \*CastToListBox()

*page 11*     virtual XVT\_ListButton \*CastToListButton()

*page 11*     virtual XVT\_ListEdit \*CastToListEdit()

*page 11*     virtual XVT\_MenuWin \*CastToMenuWin()

*page 11*     virtual XVT\_PrintWin \*CastToPrintWin()

*page 11*     virtual XVT\_RadioButton \*CastToRadioButton()

*page 11*     virtual XVT\_ScreenWin \*CastToScreenWin()

*page 11*     virtual XVT\_ScrollBar \*CastToScrollBar()

*page 11*     virtual XVT\_StaticText \*CastToStaticText()

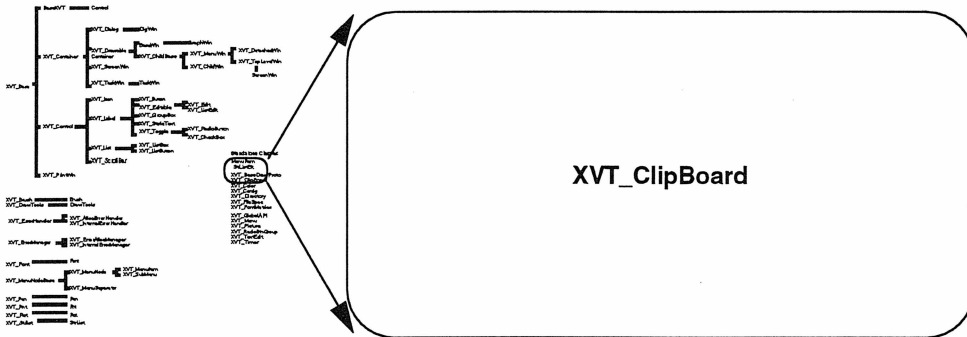
*page 11*     virtual XVT\_TaskWin \*CastToTaskWin()

*page 11*     virtual XVT\_TopLevelWin \*CastToTopLevelWin()

*page 12*     virtual XVT\_Rct GetInnerRect()

*page 13*     virtual XVT\_Rct GetOuterRect()

# XVT\_ClipBoard



## Overview

<b>Header File</b>	clipbd.h
<b>Source File</b>	clipbd.cc
<b>Superclass</b>	
<b>Subclasses</b>	
<b>Usage</b>	Concrete

An instance of the clipboard object structures access to the clipboard.

You may dynamically create an instance of this class whenever you need access to the clipboard. There is actually only one instance of the clipboard; it is created by the `task_window` when it initializes and is available as the global variable `XVT_CB`. If you attempt to create more than one instance, `operator new` returns a pointer to the first instance created.

Since instances of the `XVT_ClipBoard` do have state, an attempt to statically allocate an instance causes an error. Always use `new` to create an instance of `XVT_ClipBoard`.



## Example

Here is how you might put text data on the clipboard:

```
{
    char* myString = "This is some data for the
clipboard";
    XVT_ClipBoard* theClipboard;

    theClipboard = new XVT_ClipBoard;
    theClipboard->PutData(
        myString,
        sizeof( myString ) );
    delete theClipboard;
}
```

And here is how you could retrieve it:

```
{
    char* myString;
    long myStringLen;
    XVT_ClipBoard* theClipboard;

    theClipboard = new XVT_ClipBoard;
    if (theClipboard->FormatAvail( CB_TEXT, (char*)0 ))
    {
        myString = theClipboard->GetData( &myStringLen );
        if (myString)
        {
            // do things with the string data
            .
            .
            .

            delete myString;
        }
    }
    delete theClipboard;
}
```

## Constructors

```
XVT_ClipBoard()
~XVT_ClipBoard()
```

## Operators

```
void* operator new( size_t amount )
void operator delete( void* cb )
```

## Member Functions

---

### XVT\_ClipBoard::Close

CLOSE THE CLIPBOARD

---

#### Prototypes

```
BOOLEAN  
Close()
```

#### Return Value

TRUE if the clipboard was successfully closed, FALSE if not.

#### Description

Close the clipboard. You should make this call as soon after reading or writing data to the clipboard as possible. Since other applications may be prevented from accessing the clipboard when it is open, you should close it as soon as possible after opening it. In particular, you should not return to the main loop between a call to `Open` and a call to `Close`.

---

### XVT\_ClipBoard::FormatAvail

DETERMINE IF A CLIPBOARD FORMAT IS AVAILABLE

---

#### Prototypes

```
BOOLEAN  
FormatAvail(  
    CB_FORMAT          format,  
    const char*         name )
```

#### Parameters

**format**  
The desired clipboard format.

**name**  
A null-terminated character string of 4 characters or less that serves as the clipboard format name for application-defined (CB\_APPL) clipboard data types.

## Return Value

TRUE if the requested format is available, FALSE if not.

## Description

Tests to see if data in a particular format is on the clipboard.

Valid formats are:

### CB\_TEXT

This format consists of a sequence of ASCII characters, possibly broken into lines that are terminated with an end-of-line sequence whose value is in the constant `EOL_SEQ`. In all cases, the sequence is either a plain carriage return (`\r`), a plain line feed (`\n`), or a carriage return followed by a line feed (`\r\n`). The entire sequence is not terminated with a NULL byte. The only way to determine its end is to refer to the size parameter, which always accompanies the data itself.

When breaking `CB_TEXT` data into lines (such as after calling `GetData`), it's easiest to use the function `FindEOL`. There's no need to use `EOL_SEQ` directly. However, when building `CB_TEXT` data, you must concatenate the contents of `EOL_SEQ` onto each line (with `strcat` or `gstrcat`, for example). The last line is not required to end with an end-of-line sequence.

### CB\_PICT

This format consists of a linear sequence of bytes that represent an encapsulated picture. The internals of this format are undefined, but you may assume that the bytes can safely be passed from one address space to another (unlike a non-linearized `XVT_Picture`).

If you already have an object of type `XVT_Picture`, you can put it onto the clipboard directly with `PutData`. You do not need to linearize it first. If you get a linearized picture off the clipboard with `GetData`, you can turn it into an `XVT_Picture` object with the data version of the `XVT_Picture` constructor.

### CB\_APPL

This format lets you put your own data structures onto the clipboard, presumably for use by other applications that know about those data structures. Each format has a name, which consists of from 1 to 4 alphabetic and numeric characters. When referring to a `CB_APPL` format, you must also specify the name. You can put as many `CB_APPL` formats onto the clipboard as you want (along with `CB_TEXT` and `CB_PICT` formats, if you like), as long as they have different names.

The only requirement placed on your CB\_APPL data structures is that they must be address-space independent, since they may be passed from one application to another. This means that they must not contain pointers, because those pointers will be invalid to the receiving application. Another way to think about whether a data structure is valid is to ask yourself whether, if it were written to a file, it could be read back in and properly interpreted at a later time by another instance of your formats.

The clipboard need not be opened with Open to call FormatAvail.

### Equivalent C Function

cb\_format\_avail()

---

## XVT\_ClipBoard::GetData

RETRIEVE CLIPBOARD DATA

---

### Prototypes

```
char*
GetData(
    CB_FORMAT          format
    const char*        name
    long*              size ) const
```

### Parameters

**format**  
The desired clipboard format.

**name**  
A null-terminated character string of 4 characters or less that serves as the clipboard format name for application-defined (CB\_APPL) clipboard data types.

**size**  
The size of the data pointed to by the returned pointer.

### Return Value

The clipboard data or NULL if the requested format was not available. The returned pointer points to memory allocated by the clipboard. You will have to dispose of it using dispose. For the CB\_PICT format, the returned data is a picture object that you will have to dispose of.

**Description**

GetData( name, size )  
Retrieves clipboard data in the CB\_APPL format.

GetData( size )  
Retrieves clipboard data in the CB\_TEXT format.

GetData( boundary )  
Retrieves clipboard data in the CB\_PICT format.

**Equivalent C Function**

cb\_open()  
cb\_close()  
cb\_malloc()  
cb\_free()  
cb\_get()

---

**XVT\_ClipBoard::GetOpenState**

DETERMINE IF THE CLIPBOARD IS CURRENTLY OPEN

---

**Prototypes**

BOOLEAN  
GetOpenState() const

**Return Value**

TRUE if the clipboard is currently open, FALSE if not.

---

**XVT\_ClipBoard::Open**

OPEN THE CLIPBOARD

---

**Prototypes**

BOOLEAN  
Open(                      BOOLEAN                      writing )

**Parameters**

writing  
A flag which is TRUE if the clipboard is to be opened for writing,  
FALSE if for reading.

**Return Value**

TRUE if the clipboard was successfully opened, FALSE if not.

**Description**

Prepare the clipboard for reading or writing data. You should make this call immediately before reading or writing data to the clipboard.

---

## XVT\_ClipBoard::PutData

PUT DATA ON THE CLIPBOARD

---

**Prototypes**

```

BOOLEAN
PutData(
    void*          data,
    const char*    text,
    long           size )

BOOLEAN
PutData(
    const char*    text,
    long           size )

BOOLEAN
PutData(
    XVT_Picture*   pict )

```

**Parameters**

**name**  
A null-terminated character string of 4 characters or less that serves as the clipboard format name for application-defined (CB\_APPL) clipboard data types.

**data**  
pointer to the data to be placed on the clipboard.

**size**  
The size in bytes of the data at which data points.

**pict**  
The picture to be placed on the clipboard.

**Return Value**

TRUE if the operation was successful, FALSE if not.

**Description**

```
void PutData( name, data, size )  
    Puts CB_APPL data on the clipboard.  
void PutData( data, size )  
    Puts CB_TEXT data on the clipboard.  
void PutData( pict )  
    Puts picture data on the clipboard.
```

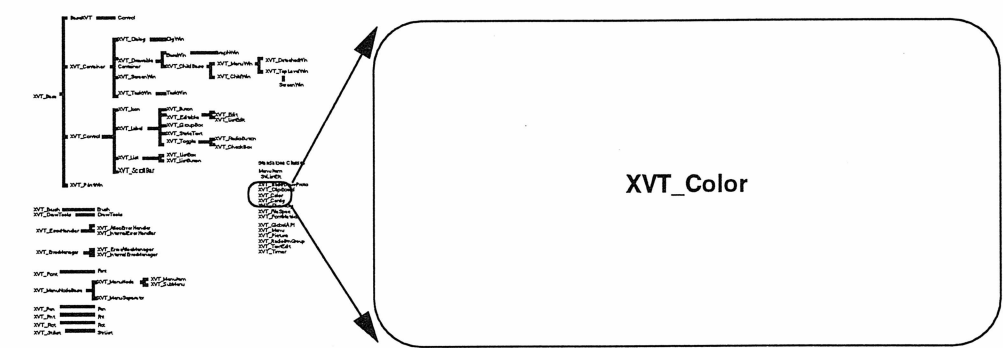
**Equivalent C Function**

```
cb_open()  
cb_close()  
cb_malloc()  
cb_free()  
cb_put()
```

**Implementation Members**

```
Mem  
OpenState  
RefCount
```

# XVT\_Color



## Overview

Header File	tools.h
Source File	tools.cc
Superclass	
Subclasses	
Usage	Concrete

Instances of XVT\_Color represent RGB colors.

## Constructors

```
XVT_Color(  
    unsigned short red = 0,  
    unsigned short green = 0,  
    unsigned short blue = 0 )  
XVT_Color( const XVT_Color& color )
```

The following macros provide pre-defined XVT\_Color objects:

```
XVT_COLOR_RED  
XVT_COLOR_GREEN  
XVT_COLOR_BLUE  
XVT_COLOR_CYAN  
XVT_COLOR_MAGENTA
```



```

XVT_COLOR_YELLOW
XVT_COLOR_BLACK
XVT_COLOR_DKGRAY
XVT_COLOR_GRAY
XVT_COLOR_LTGRAY
XVT_COLOR_WHITE

```

The RGB values for each correspond to the name of the macro. Thus, the RGB values of XVT\_COLOR\_RED are ( 0xFF, 0x00, 0x00). These macros can be wherever an object of type XVT\_Color is required. They are defined in the header **tools.h**.

```
XVT_Color (COLOR C)
```

```
~XVT_Color()
```

## Operators

```
XVT_Color& operator=( const XVT_Color& color )
```

```
BOOLEAN operator==( const XVT_Color& color )
```

Colors can be assigned and compared.

## Member Functions

---

### XVT\_Color::GetBlue

RETRIEVE THE BLUE COMPONENT OF A COLOR

---

#### Prototypes

```

unsigned short
GetBlue() const

```

#### Return Value

The blue component, a number from 0 to 255 where 0 is black.

---

### XVT\_Color::GetGreen

RETRIEVE THE GREEN COMPONENT OF A COLOR

---

#### Prototypes

```

unsigned short
GetGreen() const

```

**Return Value**

The green component, a number from 0 to 255 where 0 is black.

---

**XVT\_Color::GetRed**

RETRIEVE THE RED COMPONENT OF A COLOR

---

**Prototypes**

```
unsigned short  
GetRed() const
```

**Return Value**

The red component, a number from 0 to 255 where 0 is black.

---

**XVT\_Color::SetBlue**

SET THE BLUE COMPONENT OF A COLOR

---

**Prototypes**

```
void SetBlue(  
    unsigned short    b )
```

**Parameters**

b

The blue component, a number from 0 to 255 where 0 is black.

**Description**

Sets the blue component of a color.

---

**XVT\_Color::SetGreen**

SET THE GREEN COMPONENT OF A COLOR

---

**Prototypes**

```
void  
SetGreen(  
    unsigned short    g )
```

**Parameters**

`g`  
The green component, a number from 0 to 255 where 0 is black.

**Description**

Sets the green component of a color.

---

## XVT\_Color::SetRed

SET THE RED COMPONENT OF A COLOR

---

**Prototypes**

```
void  
SetRed(  
    unsigned short    r )
```

**Parameters**

`r`  
The red component, a number from 0 to 255 where 0 is black.

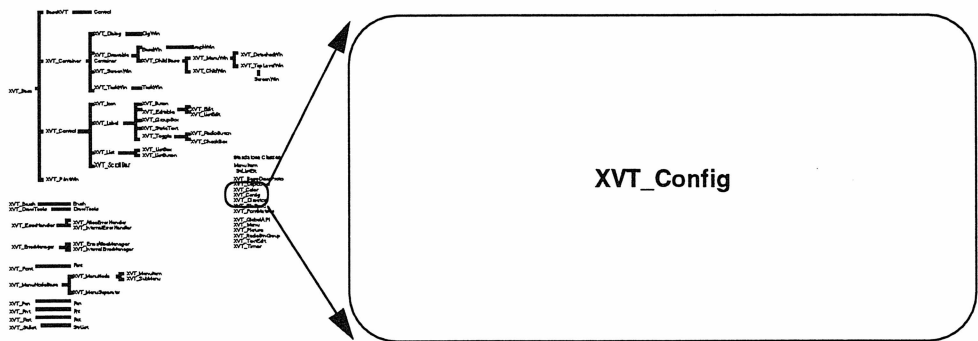
**Description**

Sets the red component of a color.

## Implementation Members

```
ConvertTo  
ConvertFrom  
RedValue  
GreenValue  
BlueValue
```

# XVT\_Config



## Overview

Header File	config.h
Source File	config.cc
Superclass	
Subclasses	
Usage	Concrete

Instances of this class provide configuration information to an XVT++ application.

## Constructors

```
XVT_Config(  
    short menubar_id = MENU_BAR_RID,  
    short about_id = 0,  
    const char* base_appl_name = "untitled",  
    const char* appl_name = "untitled",  
    const char* task_title = "untitled" )  
XVT_Config( const XVT_Config& config )  
virtual ~XVT_Config()
```

## Member Functions

---

### XVT\_Config::GetAboutBoxID

RETRIEVE THE RESOURCE ID FOR THE ABOUT BOX DIALOG

---

#### Prototypes

```
short  
GetAboutBoxID() const
```

#### Return Value

The resource ID for the about box dialog.

#### Equivalent C Function

The XVT\_CONFIG structure as declared in the application's main.

---

### XVT\_Config::GetAppName

RETRIEVE THE APPLICATION'S NAME

---

#### Prototypes

```
BOOLEAN  
GetAppName(  
    char*          buffer,  
    unsigned long* len ) const
```

#### Parameters

**buffer**  
Storage to receive the application name.

**len**  
A pointer to the length of buffer.

#### Return Value

TRUE if the length of buffer was sufficient to hold the application's name, FALSE if not. If FALSE is returned, len is set to the required length.

---

## XVT\_Config::GetBaseAppName

RETRIEVE THE BASENAME OF THE APPLICATION

---

### Prototypes

```
BOOLEAN  
GetBaseAppName(  
    char*          buffer,  
    unsigned long* len ) const
```

### Parameters

**buffer**  
Storage to receive the base application name.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the base name,  
FALSE if not. If FALSE is returned, len is set to the required length.

### Description

Retrieves the base name of the application.

---

## XVT\_Config::GetMenuBarID

RETRIEVE THE RESOURCE ID OF THE TASK MENUBAR

---

### Prototypes

```
short  
GetMenuBarID() const
```

### Return Value

The resource ID of the task menubar.

### Equivalent C Function

The XVT\_CONFIG structure as declared in the application's main.

---

## XVT\_Config::GetTaskWinTitle

RETRIEVE THE TASK WINDOW'S INITIAL TITLE

---

### Prototypes

```
BOOLEAN
GetTaskWinTitle(
    char*          buffer
    unsigned long* len ) const
```

### Parameters

**buffer**  
Storage to receive the task window's title.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the application's name, FALSE if not. If FALSE is returned, len is set to the required length.

---

## XVT\_Config::SetAboutBoxID

SET THE RESOURCE ID FOR THE ABOUT BOX DIALOG

---

### Prototypes

```
void
SetAboutBoxID(
    short          id )
```

### Parameters

**id**  
The resource ID for the about box dialog.

### Description

Sets the about box dialog resource ID. In order to take effect, this must be set before a task window is instantiated.

---

## XVT\_Config::SetAppName

SET THE APPLICATION'S NAME

---

### Prototypes

```
void  
SetAppName(  
    const char*          appl_name )
```

### Parameters

appl\_name  
The application name.

### Description

Sets the applications's name. Typically, the application name is prepended to window titles by SetDocTitle.

---

## XVT\_Config::SetBaseAppName

SET THE APPLICATION'S BASENAME

---

### Prototypes

```
void  
SetBaseAppName(  
    const char*          base_appl_nam )
```

### Parameters

base\_appl\_nam  
The base application name used to search for .hlp, .frl, and .uid files.

### Description

Sets the application's base name.



---

## XVT\_Config::SetMenuBarID

SET THE RESOURCE ID FOR THE TASK MENUBAR

---

### Prototypes

```
void
SetMenuBarID(
    short          id )
```

### Parameters

id  
The task menubar's resource ID.

### Description

Sets the task menubar resource ID. In order to take effect, this must be set before a task window is instantiated.

---

## XVT\_Config::SetTaskWinTitle

SET THE TASK WINDOW'S INITIAL TITLE

---

### Prototypes

```
void
SetTaskWinTitle(
    const char*    taskwin_title )
```

### Parameters

taskwin\_title  
The task window title.

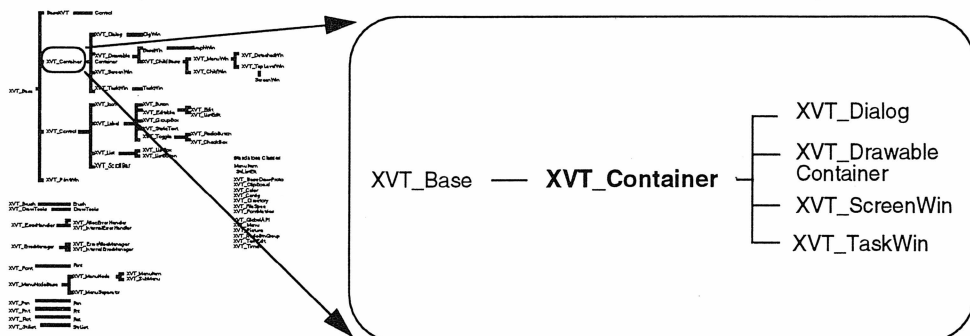
### Description

Sets the task window's initial title.

## Implementation Members

```
ConvertTo
ConvertFrom
MenuBarID
AboutBoxID
BaseApplName
ApplName
TaskWinTitle
```

# XVT\_Container



## Overview

<b>Header File</b>	contain.h
<b>Source File</b>	contain.cc
<b>Superclass</b>	XVT_Base
<b>Subclasses</b>	XVT_Dialog, XVT_DrawableContainer, XVT_ScreenWin, XVT_TaskWin
<b>Usage</b>	Implementation

The XVT\_Container class adds no interface. Its purpose is to add the protocols used to manage lists of contained windows or controls.

## Implementation Members

```

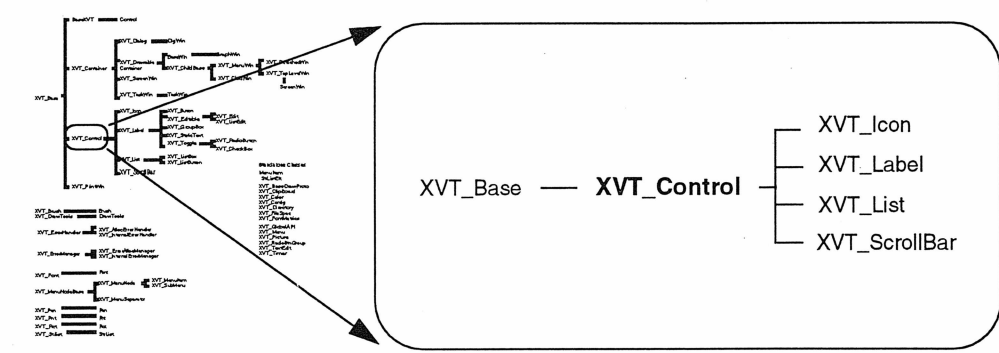
XVT_Container
~XVT_Container
GetCtlFlagsList
WinContainer
CtlContainer
CtlFlagsList
  
```

## Inherited Member Functions

### From XVT\_Base

<i>page 11</i>	<code>virtual BaseWin* CastToBaseWin()</code>
<i>page 10</i>	<code>virtual DlgWin* CastToDlgWin()</code>
<i>page 10</i>	<code>virtual ScreenWin* CastToScreenWin11()</code>
<i>page 10</i>	<code>virtual TaskWin* CastToTaskWin11()</code>
<i>page 11</i>	<code>virtual XVT_Button *CastToButton()</code>
<i>page 11</i>	<code>virtual XVT_CheckBox *CastToCheckBox()</code>
<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer*CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>

# XVT\_Control



## Overview

Header File	control.h
Source File	control.cc
Superclass	XVT_Base
Subclasses	XVT_Icon, XVT_Label, XVT_List, XVT_ScrollBar
Usage	Implementation

The XVT\_Control class defines the interface common to all controls.

## Member Functions

### XVT\_Control::Close

SCHEDULE A CONTROL FOR DESTRUCTION

#### Prototypes

```
virtual void  
Close()
```

## Description

Schedules the destruction of this control.

Do not release resources that you have attached to the control until the `e_destroy` event handler member function is called. Until `e_destroy` is called to notify the application of the control's destruction, other events can still arrive even after `Close` has been called.

After the call to the `e_destroy` event handler member function, the control object is deleted automatically. You do not need to delete it.

## Equivalent C Function

`close_window()`

---

## XVT\_Control::e\_create

RECEIVE NOTIFICATION OF A CONTROL'S CREATION

---

### Prototypes

```
virtual void  
e_create()
```

### Description

This member function must be overridden by a control subclass if the application wishes to take any actions in response to a control's creation.

This is the first event handling member function that is called in a control's lifetime. Once this function is called, the control is completely operable and the `e_create` member function of the parent (container) window will already have been called.

---

## XVT\_Control::e\_destroy

RECEIVE NOTIFICATION OF A CONTROL'S IMPENDING DESTRUCTION

---

### Prototypes

```
virtual void  
e_destroy()
```

## Description

This member function must be overridden by a control subclass if the application wishes to take any actions in response to a control's destruction.

This is the last event handling member function that is called in a control's lifetime. Once this function is called *none* of the control interface provided by XVT++ can be used. The only purpose of this call is to allow a control to de-allocate resources before it is destroyed.

---

## XVT\_Control::e\_user

RECEIVE NOTIFICATION OF A USER-DEFINED EVENT

---

### Prototypes

```
virtual long
e_user(      long          id,
            void*          data )
```

### Parameters

*id*  
The ID of the user-defined event.

*data*  
The data associated with the user-defined event.

### Description

This member function must be overridden by a control subclass if the application wishes to take any actions in response to user-defined events.

User-defined events are used for two purposes. Events with IDs ranging from 0 to 32767 can be defined by applications for whatever purpose they desire. All other IDs are reserved to XVT and can be used to deliver platform-specific events under some circumstances. See the appropriate XVT platform-specific book.

Note that there is no way to enqueue a user event on the native event queue. To deliver a user event, simply call `e_user` directly.

---

## XVT\_Control::GetEnabledState

DETERMINE WHETHER A CONTROL IS ENABLED OR DISABLED

---

### Prototypes

```
BOOLEAN  
GetEnabledState() const
```

### Return Value

TRUE if the control is enabled, FALSE if not.

---

## XVT\_Control::GetID

RETRIEVE THE CONTROL'S ID

---

### Prototypes

```
long  
GetID() const
```

### Return Value

The control's ID.

### Equivalent C Function

```
get_ctl_window()
```

---

## XVT\_Control::GetParent

RETRIEVE A CONTROL'S PARENT WINDOW

---

### Prototypes

```
XVT_Base*  
GetParent() const
```

### Return Value

The control's parent (container) window or dialog.

### Equivalent C Function

```
get_parent()
```

---

## XVT\_Control::GetVisibleState

DETERMINE IF A CONTROL IS VISIBLE

---

### Prototypes

```
BOOLEAN  
GetVisibleState() const
```

### Return Value

A flag that is TRUE if the control is visible, FALSE if not.

---

## XVT\_Control::Init

INITIALIZE A CONTROL

---

### Prototypes

```
virtual BOOLEAN  
Init()
```

### Return Value

Always TRUE because the underlying control must already exist.

### Description

Creates the native control. This version of `Init` is only for controls that have been created from resources. Instead of actually creating the native control, it just hooks the control object up with the existing native control.

### Equivalent C Function

```
create_def_control()  
create_control()
```

---

## XVT\_Control::MakeFront

GIVE A CONTROL KEYBOARD FOCUS

---

### Prototypes

```
void  
MakeFront()
```



**Description**

Give a control the keyboard focus and make it the current control with respect to future keyboard navigation requests. If the focus actually changes, the appropriate `e_focus` member functions will be called.

The abstract control classes (`XVT_Button`, `XVT_CheckBox`, and so on) provide an additional version of `Init()` for creating controls at run time.

**Equivalent C Function**

`set_front_window()`

---

## XVT\_Control::SetEnabledState

ENABLE OR DISABLE A CONTROL

---

**Prototypes**

```
void  
SetEnabledState(  
    BOOLEAN                state )
```

**Parameters**

`state`  
A flag that is TRUE if the control is to be enabled, FALSE if it is to be disabled.

**Description**

Enables or disables a control according to the `state` parameter. When a control is disabled none of its event handler member functions are called; mouse and character input instead is directed to the control's container.

**Equivalent C Function**

`enable_window()`

---

## XVT\_Control::SetInnerRect

SET A CONTROL'S DIMENSIONS

---

### Prototypes

```
void  
SetInnerRect(  
    XVT_Rct                boundary )
```

### Parameters

**boundary**  
The control's new dimensions relative to its parent windows client area.

### Description

Sets a control's dimensions. For drop-down controls, the dimensions set are the dimensions of the control when *not* dropped down.

### Equivalent C Function

`move_window()`

---

## XVT\_Control::SetVisibleState

MAKE A CONTROL VISIBLE OR INVISIBLE

---

### Prototypes

```
void  
SetVisibleState(  
    BOOLEAN                state )
```

### Parameters

**state**  
A flag that is TRUE if the control is to be made visible, FALSE if it is to be made invisible.

### Description

Makes a control visible or invisible.

### Equivalent C Function

`show_window()`

## Implementation Members

XVT\_Control  
 ~XVT\_Control  
 Parent  
 ID  
 Type  
 EnableProtocol  
 ShowProtocol  
 MoveProtocol  
 CloseProtocol  
 EnabledState  
 VisibleState

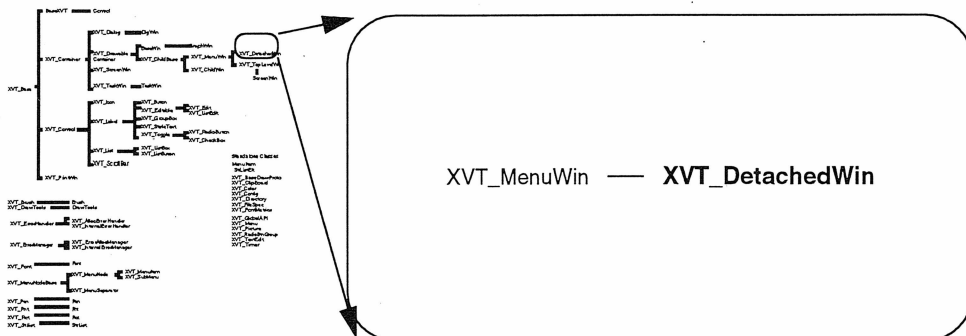
## Inherited Member Functions

### From XVT\_Base

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()

*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT DetachedWin



## Overview

<b>Header File</b>	detached.h
<b>Source File</b>	detached.cc
<b>Superclass</b>	XVT_MenuWin
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The `XVT_DetachedWin` class specifies the interface to the class of windows that can contain controls or child windows and that are *not* contained by the task window if the native window system has a task window. This class thus differs from `XVT_TopLevelWin` only under `XVT/Win` or `XVT/PM`.

You use this class by creating a subclass that overrides virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

**XVT\_DetachedWin()**

Create a detached window. The actual method by which the native window is created is determined by which Init function is called.

**virtual ~XVT\_DetachedWin()**

Removes the detached window from the screen's list of child windows.

## Member Functions

---

### XVT\_DetachedWin::Init

INITIALIZE THE WINDOW

---

#### Prototypes

**BOOLEAN**

**Init(**

WIN\_TYPE

XVT\_Rct

const char\*

long

long

wtype,

boundary,

title,

menu\_rid,

flags )

**BOOLEAN**

**Init(**

long

rid )

#### Parameters

**wtype**

The type of window to be created. It should be one of W\_DOC, W\_DBL, or W\_PLAIN.

**boundary**

The bounding rectangle (in pixels) of the window's client area. The rectangle is in screen coordinates.

**title**

The window's title. If the wtype is W\_DOC, the title is set as though SetDocTitle had been called; otherwise, it is set as though SetTitle was called.

**menu\_rid**

The resource ID for the window's menu.

**flags**

A bitwise OR'd combination of flags that control the window's attributes and decoration.

**rid**

The resource ID by means of which the window's dimensions, attributes, and contents can be located.

**Return Value**

TRUE if the window was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by windows. Recovery can be attempted by disposing of the new window, closing another window and retrying the creation of the window.

**Description**

The Init member functions create the native window and call the window's e\_create method. When execution returns from the Init call, the window is complete and ready to use. Prior to the Init call, the window is not usable.

Init( wtype, boundary, title, menu\_rid, flags )

Creates only a window with the given parameters. XVT++ control objects must be created separately by the user.

Init( rid )

Creates a window and contained controls from a resource specification. XVT++ control objects corresponding to the controls described in the resource must be created and installed separately by the application developer. The recommended place to do this is in the window's e\_create member function; however, you can create the control objects at any time. Events intended for controls that have no corresponding XVT++ control object cause a run-time error.

**Equivalent C Function**

create\_window()

create\_def\_window()

create\_res\_window()

**Implementation Members**

BOOLEAN Init( XVT\_WindowDef\* def )

## Inherited Member Functions

### From XVT\_MenuWin

- page 286*    virtual void e\_close()
- page 287*    virtual void e\_font( XVT\_Font font, FONT\_PART part )
- page 287*    XVT\_Menu \*GetMenu()
- page 288*    void GetTitle( char \*buffer, long len )
- page 289*    void SetDocTitle( char \*str )
- page 289*    void SetFontMenu( XVT\_Font font )
- page 290*    void SetMenu( XVT\_Menu \*menu )
- page 291*    void SetTitle( char \*str )

### From XVT\_ChildBase

- page 49*    virtual void e\_hscroll( SCROLL\_CONTROL activity, short pos )
- page 49*    virtual void e\_vscroll( SCROLL\_CONTROL activity, short pos )
- page 50*    XVT\_TextEdit\* GetActiveTextEdit()
- page 50*    XVT\_Pnt GetCaretPos() const
- page 51*    BOOLEAN GetCaretState() const
- page 51*    BOOLEAN GetEnabledState()
- page 51*    XVT\_ChildBase \*GetParent() const
- page 52*    long GetScrollPosition( SCROLL\_TYPE scroll\_type ) const
- page 52*    long GetScrollProportion( SCROLL\_TYPE scroll\_type ) const
- page 53*    void GetScrollRange( SCROLL\_TYPE scroll\_type, long \*min, long \*max ) const
- page 54*    XVT\_TextEdit\* GetTextEdit( long id )
- page 54*    BOOLEAN GetVisibleState()
- page 55*    void MakeFront()
- page 55*    void ReleaseMouse()
- page 56*    void SetCaretDimensions( XVT\_Pnt vector )



<i>page 56</i>	<code>void SetCaretPos( XVT_Pnt point )</code>
<i>page 57</i>	<code>void SetCaretState( BOOLEAN state )</code>
<i>page 57</i>	<code>void SetCursor( CURSOR cursor )</code>
<i>page 58</i>	<code>void SetEnabledState( BOOLEAN state )</code>
<i>page 59</i>	<code>void SetScrollPosition( SCROLL_TYPE scroll_type, long position )</code>
<i>page 60</i>	<code>void SetScrollProportion( SCROLL_TYPE scroll_type, long proportion )</code>
<i>page 60</i>	<code>void SetScrollRange( SCROLL_TYPE scroll_type, long min, long max, long pos )</code>
<i>page 61</i>	<code>void SetVisibleState( BOOLEAN f )</code>
<i>page 62</i>	<code>void TrapMouse()</code>

### **From XVT\_DrawableContainer**

<i>page 129</i>	<code>void Clear()</code>
<i>page 129</i>	<code>void Clear( XVT_Color color )</code>
<i>page 129</i>	<code>void Close()</code>
<i>page 128</i>	<code>XVT_BaseDrawProto* DrawProtocol</code>
<i>page 130</i>	<code>virtual void e_char( short chr, BOOLEAN shift, BOOLEAN control)</code>
<i>page 131</i>	<code>virtual void e_create()</code>
<i>page 132</i>	<code>virtual void e_destroy()</code>
<i>page 132</i>	<code>virtual void e_focus( BOOLEAN active )</code>
<i>page 133</i>	<code>virtual void e_mouse_dbl( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )</code>
<i>page 134</i>	<code>virtual void e_mouse_down( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )</code>

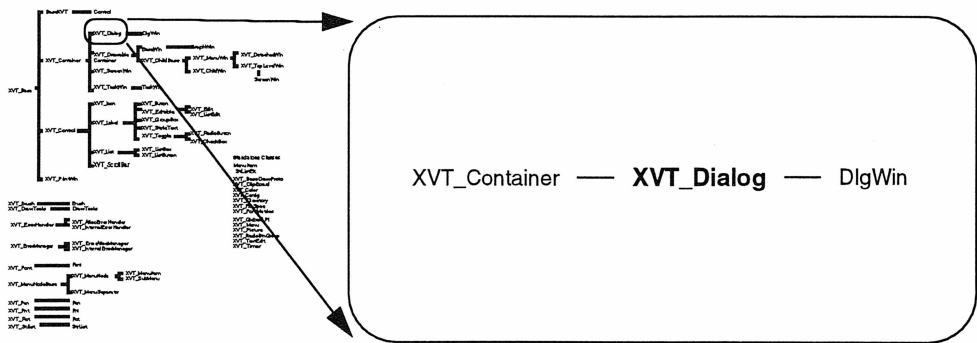
- page 135*     virtual void e\_mouse\_move(  
                 XVT\_Pnt point,  
                 BOOLEAN shift,  
                 BOOLEAN control,  
                 short button )
- page 135*     virtual void e\_mouse\_up(  
                 XVT\_Pnt point,  
                 BOOLEAN shift,  
                 BOOLEAN control,  
                 short button )
- page 136*     virtual void e\_size( XVT\_Rct boundary )
- page 137*     virtual void e\_timer( long id )
- page 137*     virtual void e\_update( XVT\_Rct boundary )
- page 139*     virtual long e\_user( long id, void \*data )
- page 140*     XVT\_Control \*GetCtl( long cid )
- page 140*     long GetCtlCount()
- page 141*     EVENT\_MASK GetEventMask() const
- page 141*     XVT\_Control \*GetFirstCtl()
- page 142*     XVT\_ChildBase \*GetFirstWin()
- page 142*     XVT\_Control \*GetNextCtl()
- page 143*     XVT\_ChildBase \*GetNextWin()
- page 143*     long GetWinCount()
- page 144*     void Invalidate()
- page 144*     void Invalidate( XVT\_Rctregion )
- page 145*     void Scroll(  
                 XVT\_Rct boundary,  
                 long dh,  
                 long dv )
- page 146*     void SetEventMask( EVENT\_MASK ask )
- page 148*     void SetInnerRect( XVT\_Rct r )

### **From XVT\_Base**

- page 11*     virtual BaseWin\* CastToBaseWin()
- page 10*     virtual DlgWin\* CastToDlgWin()
- page 10*     virtual ScreenWin\* CastToScreenWin11()

*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_Dialog



## Overview

Header File	dialog.h
Source File	dialog.cc
Superclass	XVT_Container
Subclasses	DlgWin
Usage	Concrete

The XVT\_Dialog class defines the interface of all dialogs.

You use this class by creating a subclass that overrides virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

```
XVT_Dialog()
virtual ~XVT_Dialog()
```

## Member Functions

---

### XVT\_Dialog::Close

SCHEDULE A DIALOG'S DESTRUCTION

---

#### Prototypes

```
void  
Close()
```

#### Description

Schedules the destruction of this dialog. Typically, this function is called in response to an `e_close` call or whenever the application needs to dispose of a dialog.

The dialog is notified of its impending destruction by a call to its `e_destroy` event handling member function. Do not release resources that you have attached to the dialog until the `e_destroy` event handler member function is called. Until `e_destroy` is called to notify the application of the control's destruction, other events can still arrive even after `Close` has been called.

After the call to the `e_destroy` event handler member function, the dialog object is deleted automatically. You do not need to delete it.

#### Equivalent C Function

```
close_window()
```

---

### XVT\_Dialog::e\_char

RECEIVE NOTIFICATION OF CHARACTER INPUT

---

#### Prototypes

```
virtual void e_char(  
    short          chr,  
    BOOLEAN        shift,  
    BOOLEAN        control )
```

#### Parameters

```
chr  
    The input character.
```

**shift**

A flag that is TRUE if the shift key was depressed, FALSE otherwise.

**control**

A flag that is TRUE if the control key was depressed, FALSE otherwise.

**Description**

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to a character being typed by the user.

A call to this function is generated when the user types an ASCII character or a function key. If the key is held down and auto-repeat occurs, a separate event is generated for each repetition. Repeated characters don't require special handling.

If the user types an upper-case character or an ASCII control character (such as `\t` or `\b`), the true ASCII value will be in `chr`, so it's not necessary to look at `shift` or `control` to see what was actually typed.

XVT provides a set of virtual key codes that represent non-standard characters. Test for a virtual key code, as opposed to a character, by comparing the `chr` argument against the constant `UCHAR_MAX`; values greater than `UCHAR_MAX` represent virtual keys.

You can change the mapping of raw key codes (as generated by the keyboard) to XVT virtual key codes, or add new codes, by changing the default keyboard hook function. This is done with `XVT_GlobalAPI::SetAttrValue` and the attribute `ATTR_KEY_HOOK`. For details, see the platform-specific books.

**Implementation Notes****XVT/CH**

In non-DOS environments only the shift information is available.

**XVT/Win, XVT/PM**

Control keys are normally used for accelerators and hence may not get delivered to dialogs.

**XVT/Mac**

The option key is used to generate non-ASCII characters. The character will be available in `chr` as usual but no indication that the option key was pressed is available.

---

## XVT\_Dialog::e\_close

RECEIVE NOTIFICATION OF A USER CLOSE REQUEST

---

### Prototypes

```
virtual void  
e_close()
```

### Description

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to a close request from the user.

A call to `e_close` is generated whenever the user tries to close the dialog by manipulating some sort of “close control” in the dialog border.

When this event is received, the dialog hasn’t actually been closed; your application must explicitly call `Close` to accomplish that. Additional event handler member functions (such as `e_focus`) may then be called for the dialog, and your application must be prepared to handle them. The last event handler member function called for a dialog will be `e_destroy`.

If the `e_close` implementation does not call `Close`, then the dialog is not closed, and nothing in the application changes. This distinction is important. Typically, a dialog checks its state when `e_close` is called. If the state indicates that the contents of the dialog have been saved (for example), then the application can simply call `Close`. If, however, the contents have not been saved, the application may display a dialog asking if the user wishes to save or discard changes, so that the changes may be preserved before the call to `Close` is made.

---

## XVT\_Dialog::e\_create

RECEIVE NOTIFICATION OF DIALOG CREATION

---

### Prototypes

```
virtual  
void e_create()
```

## Description

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to a dialog's creation.

This is the first event handling member function that is called in a dialog's lifetime. When this function is called, the dialog is completely operable but none of its controls will have been instantiated. Initial operations on controls should thus be performed when the *control's* e\_create is called.

---

## XVT\_Dialog::e\_destroy

RECEIVE NOTIFICATION OF A DIALOG'S IMPENDING DESTRUCTION

---

### Prototypes

```
virtual void  
e_destroy()
```

### Description

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to a dialog's destruction.

This is the last event handling member function that is called in a control's lifetime. Once this function is called *none* of the dialog interface provided by XVT++ can be used. The only purpose of this call is to allow a dialog to de-allocate its resources before it is destroyed.

---

## XVT\_Dialog::e\_focus

RECEIVE NOTIFICATION OF KEYBOARD FOCUS CHANGE

---

### Prototypes

```
virtual void  
e_focus(  
    BOOLEAN                active )
```



**Parameters****active**

A flag that is TRUE if the dialog is gaining focus and FALSE if it is losing focus.

**Description**

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to focus changes involving the dialog.

Calls to this member function notify the application that a dialog has either gained or lost the keyboard focus. (These conditions are known as activation and deactivation.) This may have been triggered by the user selecting a window or dialog (thus moving the focus), or by the application via a member function call, such as `MakeFront`. In either case, the application is notified that the focus has been changed.

For a given dialog, a call to `e_focus( TRUE )` is always guaranteed to be paired with either a subsequent call to `e_focus( FALSE )` or, if the window has been closed, a call to `e_destroy` member function. Deactivation events are always followed by activation events, and vice versa, until the window has been closed.

---

## XVT\_Dialog::e\_size

RECEIVE NOTIFICATION OF A SIZE CHANGE

---

**Prototypes**

```
virtual void
e_size(
    XVT_Rct                boundary )
```

**Parameters****boundary**

The dialog's new dimensions.

**height**

The dialog's new height.

**Description**

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to size changes involving the dialog.

This member function is called under several circumstances:

*dialog creation*

A call to `e_size` is generated immediately after the call to `e_create`.

*user resizes*

A call to `e_size` is generated whenever the user resizes a dialog using the border controls.

*application resizes*

A call to `e_size` is generated whenever the application resizes a dialog using `SetInnerRect`.

Use the new size information in boundary to logically rearrange or scale the dialog contents. If your application adjusts controls to fit the new size, it should be done while processing this event.

---

## XVT\_Dialog::e\_timer

RECEIVE NOTIFICATION OF TIMER EXPIRATION

---

### Prototypes

```
virtual void  
e_timer(  
    XVT_Timer*          timer )
```

### Parameters

timer  
The timer that expired.

### Description

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to timer expirations.

Timers are established by creating an instance of `XVT_Timer` and removed by deleting that instance. It is not necessary to reset the timer. It will generate calls to `e_timer` at the desired interval until it is destroyed.

---

## XVT\_Dialog::e\_user

RECEIVE NOTIFICATION OF A USER-DEFINED EVENT

---

### Prototypes

```
virtual long  
e_user(  
    long                id,  
    void*               data )
```

### Parameters

**id**  
The ID of the user-defined event.

**data**  
The data associated with the user-defined event.

### Description

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to user-defined events.

User-defined events are used for two purposes. Events with IDs ranging from 0 to 32767 can be defined by applications for whatever purpose they desire. All other IDs are reserved to XVT and can be used to deliver platform-specific events under some circumstances. See the platform-specific books.

Note that there is no way to enqueue a user event on the native event queue. To deliver a user event, simply call `e_user` directly.

---

## XVT\_Dialog::GetCtl

RETRIEVE A CONTROL BY CONTROL ID

---

### Prototypes

```
XVT_Control*  
GetCtl(  
    long                cid )
```

### Parameters

**cid**  
A control ID.

**Return Value**

The control object associated with the control ID given by cid.

---

**XVT\_Dialog::GetCtlCount**

RETRIEVE THE NUMBER OF CONTROLS IN A DIALOG

---

**Prototypes**

```
long  
GetCtlCount() const
```

**Return Value**

The number of controls in a dialog.

---

**XVT\_Dialog::GetEnabledState**

DETERMINE WHETHER A DIALOG IS ENABLED

---

**Prototypes**

```
BOOLEAN  
GetEnabledState() const
```

**Return Value**

TRUE if the dialog is enabled, FALSE if not.

---

**XVT\_Dialog::GetEventMask**

RETRIEVE THE CONTAINER'S EVENT MASK

---

**Prototypes**

```
EVENT_MASK  
GetEventMask() const
```

**Return Value**

The current event mask.

**Equivalent C Function**

```
get_event_mask()
```

---

## XVT\_Dialog::GetFirstCtl

RETRIEVE THE FIRST CONTROL IN A DIALOG

---

### Prototypes

```
XVT_Control*
GetFirstCtl()
```

### Return Value

The first control in the dialog or NULL if there were no controls.

### Description

Retrieves the first control in the list of controls and sets the control list traversal context such that subsequent calls to `GetNextControl` will retrieve subsequent controls. The entire list of controls in an object definition can be traversed by using the following code:

```
theControl = myDlg->GetFirstControl();
do
{
// ...whatever...
}
while (theControl = myDlg->GetNextControl())
```

---

## XVT\_Dialog::GetNextCtl

RETRIEVE SUBSEQUENT CONTROLS IN A DIALOG

---

### Prototypes

```
XVT_Control*
GetNextCtl()
```

### Return Value

The next control in the entry list or NULL if the end of the control list has been reached.

---

## XVT\_Dialog::GetTitle

RETRIEVE A DIALOG'S TITLE

---

### Prototypes

```
BOOLEAN  
GetTitle(  
    char*          buffer,  
    unsigned long* len ) const
```

### Parameters

**buffer**  
Storage to receive the dialog's title.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the application's name, FALSE if not. If FALSE is returned, len is set to the required length.

### Equivalent C Function

get\_title()

---

## XVT\_Dialog::GetVisibleState

DETERMINE IF A DIALOG IS VISIBLE

---

### Prototypes

```
BOOLEAN  
GetVisibleState() const
```

### Return Value

TRUE if the dialog is visible, FALSE if not.

---

## XVT\_Dialog::Init

INITIALIZE A DIALOG

---

### Prototypes

```
BOOLEAN  
Init(  
    long  
    rid )
```

### Parameters

**rid**  
The resource ID by means of which the dialog's dimensions, attributes, and contents may be located.

### Return Value

TRUE if the dialog was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by dialogs. Recovery can be attempted by disposing of the new dialog, closing another dialog, and retrying the creation of the dialog.

### Description

The Init member functions create the native dialog and call the dialog's e\_create method. When execution returns from the Init call, the dialog is complete and ready to use. Prior to the Init call, the dialog is not usable.

**Init( rid )**  
Creates a dialog and contained controls from a resource specification. XVT++ control objects corresponding to the controls described in the resource must be created and installed separately by the application developer. The recommended place to do this is in the dialog's e\_create member function; however, you can create the control objects at any time. Events intended for controls that have no corresponding XVT++ control object will cause a run-time error.

### Equivalent C Function

```
create_def_dialog()  
create_res_dialog()
```

---

## XVT\_Dialog::SetEnabledState

ENABLE OR DISABLE A DIALOG

---

### Prototypes

```
void  
SetEnabledState(  
    BOOLEAN                state )
```

### Parameters

state  
A flag that is TRUE if the dialog is to be enabled, FALSE if it is to be disabled.

### Description

Enables or disables a dialog according to the state parameter. When a dialog is disabled, its e\_focus and e\_char event handler member functions are not called and those events are directed to the dialog's parent.

### Equivalent C Function

```
enable_window()
```

---

## XVT\_Dialog::SetEventMask

SET THE CONTAINER'S EVENT MASK

---

### Prototypes

```
void  
SetEventMask(  
    EVENT_MASK            mask )
```

### Parameters

mask  
The new event mask.

### Description

Sets the container's event mask. The event mask is a bitwise OR'd combination of masks, one for each type of event. If the mask bit is set, the corresponding event handler member function is called when that type of event occurs; otherwise, the event is ignored. In some cases applications run more efficiently if undesired events are



masked off rather than just ignored by the application. Valid event masks may be constructed by ORing together the following constants:

EM\_NONE

No event handling member functions are called.

EM\_ALL

All event handling member functions are called.

EM\_CREATE

e\_create is called iff (if and only if) set.

EM\_DESTROY

e\_destroy is called iff set.

EM\_FOCUS

e\_focus is called iff set.

EM\_SIZE

e\_size is called iff set.

EM\_UPDATE

e\_update is called iff set.

EM\_CLOSE

e\_close is called iff set.

EM\_CHAR

e\_char is called iff set.

EM\_CONTROL

Control e\_action member functions of contained controls is called if set.

EM\_TIMER

e\_timer is called iff set.

EM\_USER

e\_user is called iff set.

### Equivalent C Function

set\_event\_mask()

---

## XVT\_Dialog::SetInnerRect

SET A DIALOG'S SIZE AND POSITION

---

### Prototypes

```
void
SetInnerRect(
    XVT_Rct                boundary )
```

**Parameters**

boundary

The rectangle giving the new coordinates of the dialog's client area relative to the task window, or relative to the screen if the native window system has no task window.

**Description**

This function moves and/or resizes a dialog such that its client rectangle has the coordinates given in boundary.

**Implementation Notes**

XVT/XM

The window manager may choose not to honor a move request.

**Equivalent C Function**

move\_window()

---

## XVT\_Dialog::SetTitle

SET A DIALOG'S TITLE

---

**Prototypes**

```
void  
SetTitle(  
    const char*      str )
```

**Parameters**

str  
The new title.

**Description**

Sets a dialog's title.

**Equivalent C Function**

set\_title()

---

## XVT\_Dialog::SetVisibleState

MAKE A DIALOG VISIBLE OR INVISIBLE

---

### Prototypes

```
void  
SetVisibleState(  
    BOOLEAN                state )
```

### Parameters

state  
A flag that is TRUE if the dialog is to be visible, FALSE if it is to be invisible.

### Description

This function makes a dialog visible or invisible. An invisible dialog does not appear on the screen and cannot have focus or receive input events. If a dialog with focus is made invisible, focus is transferred to another window or dialog within the application or to the task window if there are no other top level windows. Since the dialog cannot receive input events, the event handler member functions `e_focus` and `e_char` are not called.

### Equivalent C Function

```
show_window()
```

## Implementation Members

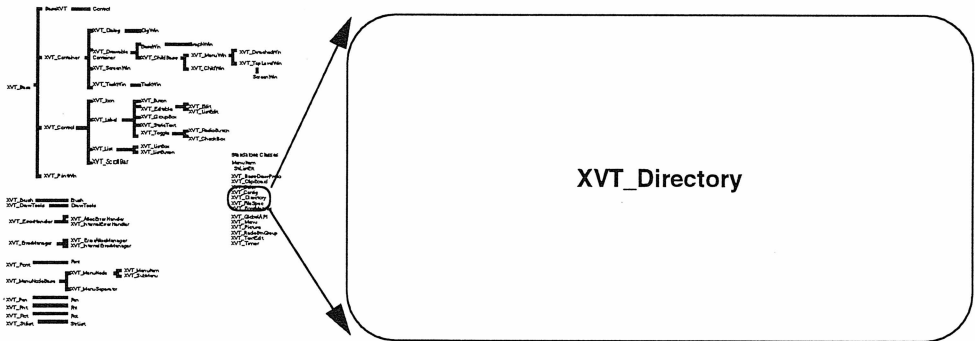
```
BOOLEAN Init( XVT_DialogDef* def )  
RemoveCtl  
Install  
TitleProtocol  
ShowProtocol  
EnableProtocol  
MoveProtocol  
CloseProtocol  
EnabledState  
VisibleState  
ControlEvent
```

## Inherited Member Functions

### From XVT\_Base

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_Directory



# Overview

<b>Header File</b>	filespec.h
<b>Source File</b>	filespec.cc
<b>Superclass</b>	
<b>Subclasses</b>	
<b>Usage</b>	Concrete

Instances of the `XVT_Directory` class represent native directories in a portable, opaque fashion.

# Constructors

XVT\_Directory()

Create an `XVT` directory object representing the current directory.

```
XVT_Directory( DIRECTORY dir )
```

```
XVT_Directory( const XVT_Directory& dir )
```

```
XVT_Directory( const char* path )
```

Create an XVT directory object representing the directory specified non-portably in `str`. Equivalent to `str_to_dir`.

```
~XVT_Directory()
```

## Operators

XVT\_Directory& operator=( const XVT\_Directory& dir )  
Directories may be assigned.

## Member Functions

---

### XVT\_Directory::DirToStr

RETRIEVE A NONPORTABLE STRING DIRECTORY SPECIFICATION

---

#### Prototypes

```
BOOLEAN  
DirToStr(  
    char*          buffer,  
    unsigned long* len )
```

#### Parameters

buffer  
Storage to receive the directory name.

len  
A pointer to the length of buffer.

#### Return Value

TRUE if the length of buffer was sufficient to hold the directory name, FALSE if not. If FALSE is returned, len is set to the required length.

#### Description

Retrieves a nonportable string directory specification suitable for passing to native functions.

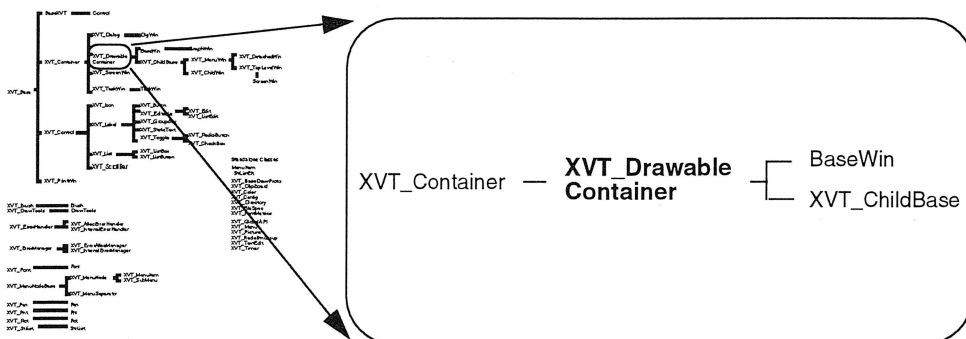
#### Equivalent C Function

dir\_to\_str()

## Implementation Members

XVT\_Directory( DIRECTORY dir )  
ConvertTo

# XVT\_DrawableContainer



## Overview

<b>Header File</b>	drawable.h
<b>Source File</b>	drawable.cc
<b>Superclass</b>	XVT_Container
<b>Subclasses</b>	BaseWin, XVT_ChildBase
<b>Usage</b>	Implementation

The drawable container class defines the interface common to all windows that can contain child-windows or controls.

Applications make use of this functionality through subclasses; they do not directly subclass or instantiate `XVT_DrawableContainer`.

## Member Variables

---

### XVT\_DrawableContainer::DrawProtocol

---

THE WINDOW'S DRAWING PROTOCOL

---

#### Prototype

```
XVT_BaseDrawProto*  
DrawProtocol
```

#### Description

The drawing protocol provides access to all of the XVT++ drawing functionality. Access to drawing functionality is indirected in this manner so that the drawing code can be made to work for both windows and print windows. In order to share drawing code, you should create a function, `DoDraw`, which will look something like this:

```
void  
DoDraw( XVT_BaseDrawProto* DP, MyContextInfo* Info )  
{  
    DP->DrawALine( ... );  
    .  
    .  
    . // draw the remainder  
}
```

The `Info` parameter provides whatever information you need to draw. You can then call `DoDraw` from both the `e_update` member function of your window and the `DrawAction` member function of your print window.

If printing is not a concern or you do not want to share drawing code, you can easily simplify access to the drawing code by adding inline member functions that duplicate the draw protocol interface to your window subclass. Alternatively, if you are willing to accept the restrictions imposed by multiple inheritance, you could just inherit from the draw protocol.



## Member Functions

---

### XVT\_DrawableContainer::Clear

CLEAR A WINDOW

---

#### Prototypes

```
void  
Clear()  
  
void  
Clear(  
    XVT_Color          color )
```

#### Parameters

color  
The background color to use.

#### Description

```
void Clear()  
    Clears a window by painting its entire client area in the  
    “standard” background color used by the native system.  
  
void Clear( color )  
    Clears a window by painting its entire client area in the given  
    color.
```

#### Equivalent C Function

```
clear_window()
```

---

### XVT\_DrawableContainer::Close

SCHEDULE A WINDOW'S DESTRUCTION

---

#### Prototypes

```
void  
Close()
```

#### Description

Schedules the destruction of this window. Typically, this function is called in response to an `e_close` call or whenever the application needs to dispose of a window.

The window is notified of its impending destruction by a call to its `e_destroy` event handling member function. Do not release resources that you have attached to the window until the `e_destroy` event handler member function is called. Until `e_destroy` is called to notify the application of the control's destruction, other events can still arrive even after `Close` has been called.

After the call to the `e_destroy` event handler member function, the window object is deleted automatically; you do not need to delete it.

### Equivalent C Function

`close_window()`

---

## XVT\_DrawableContainer::e\_char

RECEIVE NOTIFICATION OF CHARACTER INPUT

---

### Prototypes

```
virtual void e_char(  
    short          chr,  
    BOOLEAN        shift,  
    BOOLEAN        control )
```

### Parameters

`chr`  
The input character.

`shift`  
A flag that is TRUE if the shift key was depressed, FALSE otherwise.

`control`  
A flag that is TRUE if the shift key was depressed, FALSE otherwise.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to a character being typed by the user.

A call to this function is generated when the user types an ASCII character or a function key. If the key is held down and auto-repeat occurs, a separate event is generated for each repetition. Repeated characters don't require special handling.

If the user types an uppercase character or an ASCII control character (such as `\t` or `\b`), the true ASCII value will be in `chr`, so it's not necessary to look at `shift` or `control` to see what was actually typed.

XVT++ provides a set of virtual key codes that represent non-standard characters. Test for a virtual key code, as opposed to a character, by comparing the `chr` argument against the constant `UCHAR_MAX`; values greater than `UCHAR_MAX` represent virtual keys.

You can change the mapping of raw key codes (as generated by the keyboard) to virtual key codes, or add new codes, by changing the default keyboard hook function. This is done with `XVT_GlobalAPI::SetAttrValue` and the attribute `ATTR_KEY_HOOK`. For details, see the platform-specific books.

## Implementation Notes

### XVT/CH

In non-DOS environments, only the shift information is available.

### XVT/Win, XVT/PM

Control keys are normally used for accelerators and hence may not get delivered to windows.

### XVT/Mac

The option key is used to generate non-ASCII characters. The character is available in `chr` as usual but no indication that the option key was pressed is available.

---

## XVT\_DrawableContainer::e\_create

RECEIVE NOTIFICATION OF WINDOW CREATION

---

### Prototypes

```
virtual void
e_create()
```

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to a window's creation.

This is the first event handling member function that is called in a window's lifetime. When this function is called, the window is completely operable but none of its controls or child windows will have been instantiated. Initial operations on controls should thus be performed when the control or child window's `e_create` is called.

---

## XVT\_DrawableContainer::e\_destroy

RECEIVE NOTIFICATION OF A WINDOW'S IMPENDING DESTRUCTION

---

### Prototypes

```
virtual void  
e_destroy()
```

### Description

This member function must be overridden by a dialog subclass if the application wishes to take any actions in response to a window's destruction.

This is the last event handling member function that is called in a window's lifetime. Once this function is called, *none* of the window interface provided by XVT++ can be used. The only purpose of this call is to allow windows to de-allocate their resources before they are destroyed.

---

## XVT\_DrawableContainer::e\_focus

RECEIVE NOTIFICATION OF KEYBOARD FOCUS CHANGE

---

### Prototypes

```
virtual void  
e_focus(  
    BOOLEAN                active )
```

### Parameters

`active`

A flag that is TRUE if the window is gaining focus and FALSE if it is losing focus.

## Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to focus changes involving the window.

Calls to this member function notify the application that a window has either gained or lost the keyboard focus. (These conditions are known as activation and deactivation.) This may have been triggered by the user selecting a window or dialog (thus moving the focus), or by the application via a member function call (such as `MakeFront`). In either case, the application is notified that the focus has been changed.

For a given window, a call to `e_focus( TRUE )` is always guaranteed to be paired with either a subsequent call to `e_focus( FALSE )` or, if the window has been closed, a call to `e_destroy`. Deactivation events are always followed by activation events, and vice versa, until the window has been closed.

---

## XVT\_DrawableContainer::e\_mouse\_dbl

RECEIVE NOTIFICATION OF A DOUBLE CLICK

---

### Prototypes

```
virtual void e_mouse_dbl(
    XVT_Pnt      point,
    BOOLEAN      shift,
    BOOLEAN      control,
    short        button )
```

### Parameters

**point**  
The location of the mouse activity.

**shift**  
A flag that is `TRUE` if the shift key was held down during the mouse operation and `FALSE` if not.

**control**  
A flag that is `TRUE` if the control key was held down during the mouse operation and `FALSE` if not.

**button**  
The mouse button depressed, 0 to 2.

## Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to double clicks.

A mouse double click always shows up as the following sequence of event handler member function calls:

```
e_mouse_down
e_mouse_up
e_mouse_double
e_mouse_up
```

---

## XVT\_DrawableContainer::e\_mouse\_down

RECEIVE NOTIFICATION OF A MOUSE DOWN

---

### Prototypes

```
virtual void
e_mouse_down(
    XVT_Pnt          point,
    BOOLEAN          shift,
    BOOLEAN          control,
    short            button )
```

### Parameters

point

The location of the mouse activity.

shift

A flag that is TRUE if the shift key was held down during the mouse operation and FALSE if not.

control

A flag that is TRUE if the control key was held down during the mouse operation and FALSE if not.

button

The mouse button depressed, 0 to 2.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to mouse clicks.

---

## XVT\_DrawableContainer::e\_mouse\_move

RECEIVE NOTIFICATION OF MOUSE MOVES

---

### Prototypes

```
virtual void
e_mouse_move(
    XVT_Pnt          point,
    BOOLEAN          shift,
    BOOLEAN          control,
    short            button )
```

### Parameters

**point**  
The location of the mouse activity.

**shift**  
A flag that is TRUE if the shift key was held down during the mouse operation and FALSE if not.

**control**  
A flag that is TRUE if the control key was held down during the mouse operation and FALSE if not.

**button**  
The mouse button depressed, 0 to 2.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to mouse movement.

---

## XVT\_DrawableContainer::e\_mouse\_up

RECEIVE NOTIFICATION OF MOUSE UPS

---

### Prototypes

```
virtual void
e_mouse_up(
    XVT_Pnt          point,
    BOOLEAN          shift,
    BOOLEAN          control,
    short            button )
```

**Parameters****point**

The location of the mouse activity.

**shift**

A flag that is TRUE if the shift key was held down during the mouse operation and FALSE if not.

**control**

A flag that is TRUE if the control key was held down during the mouse operation and FALSE if not.

**button**

The mouse button depressed, 0 to 2.

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to mouse clicks.

---

**XVT\_DrawableContainer::e\_size**
RECEIVE NOTIFICATION OF A SIZE CHANGE

---

**Prototypes**

```
virtual void
e_size(
    XVT_Rct                boundary )
```

**Parameters****boundary**

The window's new dimensions.

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to size changes involving the dialog.

This member function is called under several circumstances:

*window creation*

A call to `e_size` is generated immediately after the call to `e_create`.

*user resizes*

A call to `e_size` is generated whenever the user resizes a window using the border controls.



*application resizes*

A call to `e_size` is generated whenever the application resizes a window using `SetInnerRect`.

Use the new size information in boundary to logically rearrange or scale the window contents. If your application adjusts child windows and controls to fit the new size, it should be done while processing this event.

---

## XVT\_DrawableContainer::e\_timer

RECEIVE NOTIFICATION OF TIMER EXPIRATION

---

### Prototypes

```
virtual void  
e_timer(  
    XVT_Timer*          timer )
```

### Parameters

`timer`  
The timer that expired.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to timer expirations.

Timers are established by creating an instance of `XVT_Timer` and removed by deleting that instance. It is not necessary to reset the timer. It will generate calls to `e_timer` at the desired interval until it is destroyed.

---

## XVT\_DrawableContainer::e\_update

RECEIVE NOTIFICATION OF WINDOW INVALIDATION

---

### Prototypes

```
virtual void  
e_update(  
    XVT_Rct          boundary )
```

## Parameters

`boundary`

The invalid area. Graphics inside the invalid area should be redrawn.

## Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to window invalidation.

A call to this member function is generated when the client area of a window must be redrawn in whole or in part. In response to this event, you should at least draw the part that needs updating. If you draw more than that, XVT++ may, for efficiency, temporarily reduce the clipping area so that only the part that needs updating is actually drawn.

Don't assume that only one call to `e_update` will be generated when a window needs to be redrawn. XVT++ may call `e_update` several different times for different areas of the window, or may combine the areas into a single bounding rectangle. You also can't make any assumptions about when `e_update` will be called; it may be called any time after `e_create`.

It is usually best to organize your application so that most, if not all, drawing occurs in `e_update` functions, rather than drawing things as you go along. That way the occurrence of an update event will be the usual case rather than the exception, and the program is likely to be simpler and more reliable. For example, when the data structure representing the contents of a window changes, don't draw the changes immediately. Instead, after making changes to the data structure, induce an update event with `Invalidate`.

Don't induce an update event when it's important to draw right away, to keep up with the user or to show animation. For example, when the user selects an object with the mouse, immediately draw whatever is required to show the selection; waiting for the update event may cause a noticeable delay.

Also, don't induce an update event when the user operates a scrollbar. The window will scroll much faster if you move some pixels already there with a call to `Scroll`, rather than repainting the entire window.

A newly created visible window always gets an update event for its entire client area shortly after being created, so it is not necessary to draw into a new window.

When you are calling `Invalidate` several times to invalidate disjoint areas of the window, it may be advantageous to call `UpdateWindow` between calls to `Invalidate`. This allows each update rectangle to be handled individually. Otherwise, the several disjoint update rectangles may be merged into a single rectangle, causing your application to update more of the screen than is needed. If you do this, take into account that there will be a recursive call to your window's event handler.

Many XVT++ member functions cannot be called during the execution of an `e_update` member function; calling these functions causes a fatal error. This is usually due to side effects that these functions produce within the context of an update event. For example, calling a function that causes an update to be generated from within the processing of a previous update event can cause endless recursion.

---

## XVT\_DrawableContainer::e\_user

RECEIVE NOTIFICATION OF A USER-DEFINED EVENT

---

### Prototypes

```
virtual long
e_user(
    long          id,
    void*         data )
```

### Parameters

`id`  
The ID of the user-defined event.

`data`  
The data associated with the user-defined event.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to user-defined events.

User-defined events are used for two purposes. Events with IDs ranging from 0 to 32767 can be defined by applications for whatever purpose they desire. All other IDs are reserved to XVT and can be

used to deliver platform-specific events under some circumstances. See the platform-specific books.

Note that there is no way to enqueue a user event on the native event queue. To deliver a user event, simply call `e_user` directly.

---

## XVT\_DrawableContainer::GetCtl

RETRIEVE THE CONTROL OBJECT ASSOCIATED WITH A CONTROL ID

---

### Prototypes

```
XVT_Control*
GetCtl(
    long                cid )
```

### Parameters

`cid`  
The control ID.

### Return Value

A pointer to the associated control, or NULL if none was found.

---

## XVT\_DrawableContainer::GetCtlCount

RETRIEVE THE NUMBER OF CONTROLS IN THIS WINDOW

---

### Prototypes

```
long
GetCtlCount() const
```

### Return Value

The number of controls in the window.

---

## XVT\_DrawableContainer::GetEventMask

RETRIEVE THE CONTAINER'S EVENT MASK

---

### Prototypes

```
EVENT_MASK  
GetEventMask() const
```

### Return Value

The current event mask.

### Equivalent C Function

```
get_event_mask()
```

---

## XVT\_DrawableContainer::GetFirstCtl

RETRIEVE THE FIRST CONTROL IN THE LIST OF CONTROLS

---

### Prototypes

```
XVT_Control*  
GetFirstCtl()
```

### Return Value

A pointer to the first control in the list of controls maintained by this window or NULL if there are no controls in the window.

### Description

Retrieves the first control in the list of controls and resets the traversal context used by `GetNextCtl` to the beginning of the control list.

You can retrieve all controls (in no particular order) by calling `GetFirstCtl` and then calling `GetNextCtl` repeatedly until it returns NULL.

---

## XVT\_DrawableContainer::GetFirstWin

RETRIEVE THE FIRST WINDOW IN THE LIST OF CHILD WINDOWS

---

### Prototypes

```
XVT_ChildBase*
GetFirstWin()
```

### Return Value

A pointer to the first window in the list of child windows maintained by this window.

### Description

Retrieves the first window in the list of child windows and resets the traversal context used by `GetNextWin` to the beginning of the window list.

You can retrieve all child windows (in no particular order) by calling `GetFirstWin` and then calling `GetNextWin` repeatedly until it returns `NULL`.

---

## XVT\_DrawableContainer::GetNextCtl

RETRIEVE THE NEXT CONTROL IN THE LIST OF CONTROLS

---

### Prototypes

```
XVT_Control*
GetNextCtl()
```

### Return Value

A pointer to the next control relative to the current traversal context, or `NULL` if we have reached the end of the list of controls.

### Description

Retrieves the next control and increments the context.

You can retrieve all controls (in no particular order) by calling `GetFirstCtl` and then calling `GetNextCtl` repeatedly until it returns `NULL`.

---

## XVT\_DrawableContainer::GetNextWin

RETRIEVE THE NEXT WINDOW IN THE LIST OF CHILD WINDOWS

---

### Prototypes

```
XVT_ChildBase*
GetNextWin()
```

### Return Value

A pointer to the next window relative to the current traversal context, or NULL if the end of the list of windows has been reached.

### Description

Retrieves the next window and increments the context.

You can retrieve all child windows (in no particular order) by calling `GetFirstWin` and then calling `GetNextWin` repeatedly until it returns NULL.

### Equivalent C Function

```
list_windows()
```

---

## XVT\_DrawableContainer::GetWinCount

RETRIEVE THE NUMBER OF CHILD WINDOWS

---

### Prototypes

```
long
GetWinCount() const
```

### Return Value

The number of child windows contained by this window.

---

## XVT\_DrawableContainer::Invalidate

INVALIDATE AN AREA OF A WINDOW

---

### Prototypes

```
void  
Invalidate()  
  
void  
Invalidate(  
    XVT_Rct                boundary )
```

### Parameters

boundary  
The area to be invalidated.

### Description

Marks an area of the window as being invalid. That area will be updated some time in the future.

This function is the preferred way to cause something to be drawn in a window.

Invalidate()  
Invalidates the entire client area of the window.

Invalidate( region )  
Invalidates the area defined by boundary.

### Implementation Notes

XVT/Win, XVT/PM  
The rectangle you are intend to invalidate should have its dimensions increased by one pixel on all sides; otherwise, pixels on the edges will not be redrawn correctly.

### Equivalent C Function

invalidate\_rect()



---

## XVT\_DrawableContainer::Scroll

SCROLL A RECTANGULAR REGION

---

### Prototypes

```
void  
Scroll(  
    XVT_Rct          boundary,  
    long             dh,  
    long             dv )
```

### Parameters

**boundary**

The boundary of the scroll area. No pixels outside of the boundary are affected by the scroll.

**dh**

Amount of horizontal scrolling in pixels. If  $dh > boundary.Width$ , the results are undefined.

**dv**

Amount of vertical scrolling in pixels. If  $dh > boundary.Height$ , the results are undefined.

### Description

Scrolls pixels inside a rectangular region.

A call to `e_update` is automatically generated for the part of the rectangle whose pixels were scrolled away. This call is made recursively, before `Scroll` returns. If the client area being scrolled is partially obscured by other windows, including child windows, then the resulting call or calls to `e_update` may encompass an area larger than just the rectangle exposed by the scrolling. Your application must not make assumptions about the calls to `e_update` that will be generated during scrolling.

This function is normally called when your application is changing the view of a document. Usually, your application keeps an internal data structure reflecting the view of the document, and part of the data structure indicates the origin of the window viewport into that document. Before you scroll a window's contents, you should first adjust your internal origin, so the recursively generated call to `e_update` event is encountered by an object whose origin has already been properly set.

If you are scrolling your window in response to an `e_vscroll` call, remember that when you receive a line up or page up event you want

to move the pixels downward so that the `dv` argument to `Scroll` is positive. When you get a line down or page down, `dv` will be negative. A similar relationship holds for calls to `e_hscroll`.

Before scrolling a window's pixels, you must ensure that the client area is valid, by calling `UpdateWindow`. This call should be made even before you change your application's internal viewport origin.

### Equivalent C Function

```
win_scroll_rect()
```

---

## XVT\_DrawableContainer::SetEventMask

SET THE CONTAINER'S EVENT MASK

---

### Prototypes

```
void  
SetEventMask(  
    EVENT_MASK          mask )
```

### Parameters

`mask`  
The new event mask.

### Description

Sets the container's event mask. The event mask is a bitwise OR'd combination of masks, one for each type of event. If the mask bit is set, the corresponding event handler member function is called when that type of event occurs; otherwise, the event is ignored. In some cases applications will run more efficiently if undesired events are masked off rather than just ignored by the application. Valid event masks may be constructed by ORing together the following constants:

`EM_NONE`

No event handling member functions will be called.

`EM_ALL`

All event handling member functions will be called.

`EM_CREATE`

`e_create` will be called iff (if and only if) set.

`EM_DESTROY`

`e_destroy` will be called iff set.

EM\_FOCUS  
e\_focus will be called iff set.

EM\_SIZE  
e\_size will be called iff set.

EM\_UPDATE  
e\_update will be called iff set.

EM\_CLOSE  
e\_close will be called iff set.

EM\_MOUSE\_DOWN  
e\_mouse\_down will be called iff set.

EM\_MOUSE\_UP  
e\_mouse\_up will be called iff set.

EM\_MOUSE\_DBL  
e\_mouse\_dbl will be called iff set.

EM\_MOUSE\_MOVE  
e\_mouse\_move will be called iff set.

EM\_CHAR  
e\_char will be called iff set.

EM\_VSCROLL  
e\_vscroll will be called iff set.

EM\_HSCROLL  
e\_hscroll will be called iff set.

EM\_COMMAND  
Menu-item action member functions of the associated menu will be called iff set.

EM\_FONT  
e\_font will be called iff set.

EM\_CONTROL  
Control e\_action member functions of contained controls will be called if set.

EM\_TIMER  
e\_timer will be called iff set.

EM\_USER  
e\_user will be called iff set.

### Equivalent C Function

set\_event\_mask()

---

## XVT\_DrawableContainer::SetInnerRect

---

SET A WINDOW'S SIZE AND POSITION

---

### Prototypes

```
void  
SetInnerRect(  
    XVT_Rct  
    boundary )
```

### Parameters

boundary  
The rectangle giving the new coordinates of the window's client area relative to the task window, or relative to the screen if the native window system has no task window.

### Description

This function moves and/or resizes a window such that its client rectangle has the coordinates given in boundary.

### Implementation Notes

XVT/XM  
The window manager may choose not to honor a move request.

### Equivalent C Function

move\_window()

## Implementation Members

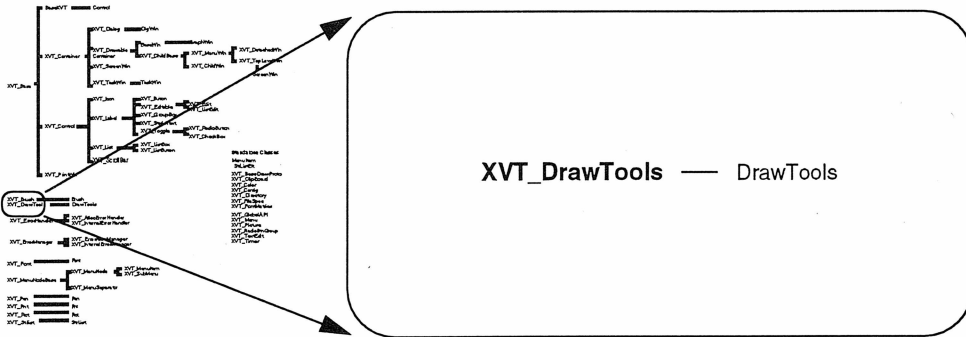
```
XVT_DrawableContainer  
~XVT_DrawableContainer  
Install  
RemoveWin  
RemoveCtl  
MoveProtocol  
CloseProtocol  
ControlEvent
```

## Inherited Member Functions

### From XVT\_Base

<i>page 11</i>	<code>virtual BaseWin* CastToBaseWin()</code>
<i>page 10</i>	<code>virtual DlgWin* CastToDlgWin()</code>
<i>page 10</i>	<code>virtual ScreenWin* CastToScreenWin11()</code>
<i>page 10</i>	<code>virtual TaskWin* CastToTaskWin11()</code>
<i>page 11</i>	<code>virtual XVT_Button *CastToButton()</code>
<i>page 11</i>	<code>virtual XVT_CheckBox *CastToCheckBox()</code>
<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer*CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>

# XVT\_DrawTools



## Overview

<b>Header File</b>	tools.h
<b>Source File</b>	tools.cc
<b>Superclass</b>	
<b>Subclasses</b>	DrawTools
<b>Usage</b>	Concrete

Instances of this class completely define how drawing primitives are rendered in a window. Each instance of `XVT_BaseDrawProto` maintains an instance of this structure as the current draw tools. The member functions `GetDrawTools` and `SetDrawTools` can be used to change the current drawing tools.

## Constructors

```

XVT_DrawTools()
XVT_DrawTools(
    XVT_Pen           pen,
    XVT_Brush         brush,
    DRAW_MODE         mode,
    XVT_Font          font,
    XVT_Color          fore_color,
    XVT_Color          back_color,
    BOOLEAN            opaque_text )
XVT_DrawTools( const XVT_DrawTools& tools )
~XVT_DrawTools()

```

## Operators

```

XVT_DrawTools& operator=( const XVT_DrawTools& tools )
    Draw tools may be assigned.

```

## Member Functions

---

### XVT\_DrawTools::GetBackColor

GET THE BACKGROUND COLOR

---

#### Prototypes

```

XVT_Color
GetBackColor() const

```

#### Return Value

A copy of the background color.

---

### XVT\_DrawTools::GetBrush

RETRIEVE THE BRUSH

---

#### Prototypes

```

XVT_Brush
GetBrush() const

```

**Return Value**

A copy of the draw tools' brush.

---

**XVT\_DrawTools::GetFont**

RETRIEVE THE FONT

---

**Prototypes**

XVT\_Font  
GetFont() const

**Return Value**

A copy of the draw tools' font.

---

**XVT\_DrawTools::GetForeColor**

GET THE FOREGROUND COLOR

---

**Prototypes**

XVT\_Color  
GetForeColor() const

**Return Value**

A copy of the draw tools' foreground color.

---

**XVT\_DrawTools::GetMode**

RETRIEVE THE DRAWING MODE

---

**Prototypes**

DRAW\_MODE  
GetMode() const

**Return Value**

The drawing mode.



---

## XVT\_DrawTools::GetOpaqueText

GET THE OPAQUE TEXT FLAG

---

### Prototypes

```
BOOLEAN  
GetOpaqueText() const
```

### Return Value

The opaque text flag.

---

## XVT\_DrawTools::GetPen

RETRIEVE THE PEN

---

### Prototypes

```
XVT_Pen  
GetPen() const
```

### Return Value

A copy of the draw tools' pen.

---

## XVT\_DrawTools::SetBackColor

SET THE BACKGROUND COLOR

---

### Prototypes

```
void  
SetBackColor(  
    XVT_Color          color )
```

### Parameters

```
color  
    The new background color.
```

**Description**

Sets the draw tools' background color.

The background color is used for the spaces between hatch marks of a patterned brush, for the text background when text is opaque, and for the background of icons.

Do not confuse the background color set by this function with any sort of automatic background painting. Your application must explicitly paint a window in the background color during a call to `e_update`, usually by calling `Clear`.

---

**XVT\_DrawTools::SetBrush**

SET THE BRUSH

---

**Prototypes**

```
void  
SetBrush(  
    XVT_Brush          brush )
```

**Parameters**

`brush`  
The new brush.

**Description**

Sets the draw tools' brush.

---

**XVT\_DrawTools::SetFont**

SET THE FONT

---

**Prototypes**

```
void  
SetFont(  
    XVT_Font          font )
```

**Parameters**

`font`  
The font that will become the draw tools' font. It should have been generated by an `e_font` call, through `GetDrawTools`, or through `GetFont`.

**Description**

Sets the draw tools' font.

**Implementation Notes**

XVT/CH

The current font is ignored. All drawing is done in whatever font the screen supports.

---

## XVT\_DrawTools::SetForeColor

SET THE FOREGROUND COLOR

---

**Prototypes**

```
void  
SetForeColor(  
    XVT_Color          color )
```

**Parameters**

color  
The new foreground color.

**Description**

Sets the draw tools' foreground color.

The foreground color is used only for drawing text and icons. Other drawing primitives take their colors from the current pen and brush.

---

## XVT\_DrawTools::SetMode

SET THE CURRENT DRAWING MODE

---

**Prototypes**

```
void  
SetMode(  
    DRAW_MODE          mode )
```

**Parameters**

mode  
The new drawing mode.

## Description

Sets the window's current drawing mode.

Drawing modes are defined by the `DRAW_MODE` enumeration, which has at least the following members:

### `M_COPY`

The normal drawing mode. The source pixels are copied to the screen, erasing any destination pixels underneath them.

### `M_XOR`

The source is XOR'd with the inverse (NOT) of the destination. This mode has the property that drawing the same thing twice is guaranteed to have no effect and that drawing something once is visible under most combinations of foreground and background colors.

### `M_OR`

The source pixels are OR'd with the destination pixels and the result is displayed on the screen.

### `M_CLEAR`

If the source pixel is set, it is written to the screen. The destination pixels are ignored.

### `M_NOT_COPY`

The inverse of the source pixels is copied to the screen.

### `M_NOT_XOR`

The inverse (NOT) of the source is XOR'd with the inverse (NOT) of the destination.

### `M_NOT_CLEAR`

If the source pixel is not set, its inverse is written to the screen. The destination pixels are ignored.

## Implementation Notes

Use of modes other than `M_COPY` for printing is not portable.

---

## XVT\_DrawTools::SetOpaqueText

SET THE OPAQUE TEXT FLAG

---

## Prototypes

```
void
SetOpaqueText(
    BOOLEAN          ot )
```

**Parameters**

ot

A flag that is TRUE if text is to be opaque and FALSE if it is to be transparent.

**Description**

If the opaque text flag is TRUE, the bounding rectangle of the text is drawn in the background color before the text itself is drawn in the foreground color.

---

## XVT\_DrawTools::SetPen

SET THE PEN

---

**Prototypes**

```
void  
SetPen(  
    XVT_Pen  
    pen )
```

**Parameters**

pen  
The new pen.

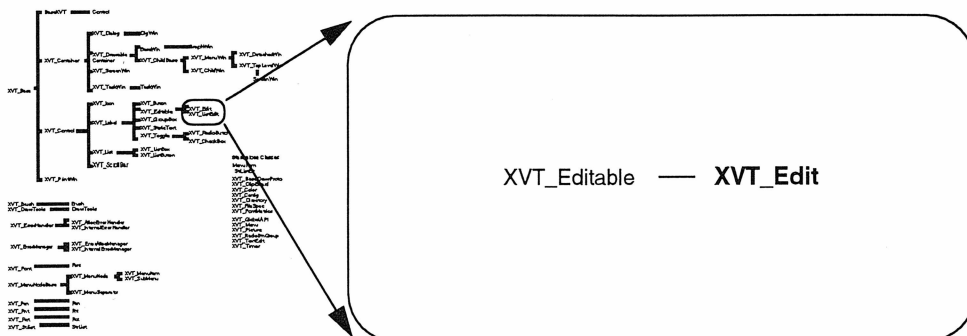
**Description**

Sets the draw tools' pen.

## Implementation Members

```
ConvertTo  
ConvertFrom  
_Pen  
_Brush  
Mode  
_Font  
ForeColor  
BackColor  
OpaqueText
```

# XVT Edit



## Overview

<b>Header File</b>	edit.h
<b>Source File</b>	edit.cc
<b>Superclass</b>	XVT_Editable
<b>Subclasses</b>	
<b>Usage</b>	Abstract

This class defines the interface to text entry field controls.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

Edit field controls allow the user to input a text string to the application. These controls vary in their appearance and behavior depending on the native GUI platform being used. For example, some systems may provide small scrollbars for these controls on one or both ends of the control. Also, platforms handle the text scrolling differently. However, these controls always report events whenever the text string is modified or the keyboard focus is gained (or lost).

XVT edit field controls are always one line high.

## Constructors

XVT\_Edit( XVT\_Dialog\* parent, long cid )

XVT\_Edit( XVT\_DrawableContainer\* parent, long cid )

## Inherited Member Functions

### From XVT\_Editable

*page 161*    virtual void e\_action()

*page 162*    e\_focus( BOOLEANactive )

*page 163*    void SelectText( long first, long last )

### From XVT\_Label

*page 239*    void GetTitle( char\* str, unsigned long\* len )

*page 239*    virtual BOOLEAN Init( XVT\_Rct boundary, long = 0L, char \*  
= NULL )

*page 240*    void SetTitle( char\* str )

### From XVT\_Control

*page 92*    virtual void Close()

*page 93*    virtual void e\_create()

*page 93*    virtual void e\_destroy()

*page 94*    virtual long e\_user( long id, void \*data )

*page 95*    BOOLEAN GetEnabledState()

*page 95*    long GetID( void )

*page 95*    XVT\_Base \*GetParent( void )

*page 96*    BOOLEAN GetVisibleState()

*page 96*    void Init()

*page 96*    void MakeFront()

*page 97*    void SetEnabledState( BOOLEAN state )

*page 98*    void SetInnerRect( XVT\_Rct boundary )

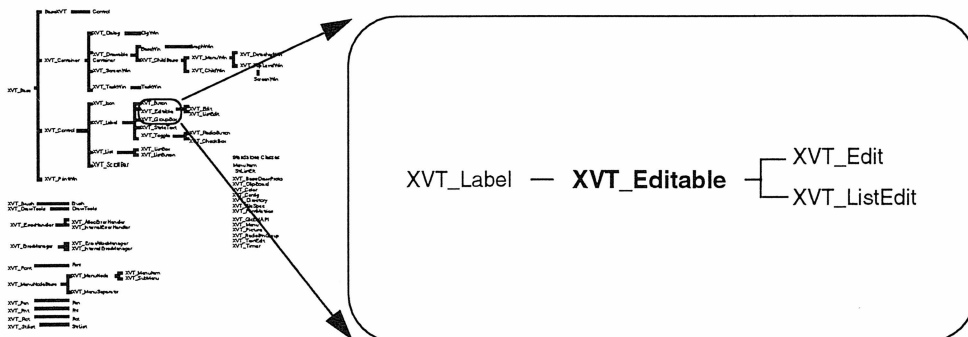
*page 98*    void SetVisibleState( BOOLEAN state )

**From XVT\_Base**

*page 11*     virtual BaseWin\* CastToBaseWin()  
*page 10*     virtual DlgWin\* CastToDlgWin()  
*page 10*     virtual ScreenWin\* CastToScreenWin11()  
*page 10*     virtual TaskWin\* CastToTaskWin11()  
*page 11*     virtual XVT\_Button \*CastToButton()  
*page 11*     virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*     virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*     virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*     virtual XVT\_Dialog \*CastToDialog()  
*page 11*     virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*     virtual XVT\_Edit \*CastToEdit()  
*page 11*     virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*     virtual XVT\_Icon \*CastToIcon()  
*page 11*     virtual XVT\_ListBox \*CastToListBox()  
*page 11*     virtual XVT\_ListButton \*CastToListButton()  
*page 11*     virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*     virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*     virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*     virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*     virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*     virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*     virtual XVT\_StaticText \*CastToStaticText()  
*page 11*     virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*     virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*     virtual XVT\_Rct GetInnerRect()  
*page 13*     virtual XVT\_Rct GetOuterRect()



# XVT Editable



## Overview

<b>Header File</b>	editable.h
<b>Source File</b>	editable.cc
<b>Superclass</b>	XVT_Label
<b>Subclasses</b>	XVT_Edit, XVT_ListEdit
<b>Usage</b>	Implementation

This class defines the interface to text edit controls.

## Member Functions

## XVT Editable::e\_action

RECEIVE NOTIFICATION OF USER ACTIVITY

## Prototypes

```
virtual void  
e_action()
```

## Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to activity in an edit field.

A call to this function is generated whenever the user modifies the contents of an edit field.

---

## XVT\_Editable::e\_focus

RECEIVE NOTIFICATION OF KEYBOARD FOCUS CHANGE

---

### Prototypes

```
virtual void  
e_focus(  
    BOOLEAN                active )
```

### Parameters

**active**  
A flag that is TRUE if the edit control is gaining focus and FALSE if it is losing focus.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to focus changes involving the edit field.

Calls to this member function notify the application that an edit control has either gained or lost the keyboard focus.

For a given edit control, a call to `e_focus( TRUE )` is always guaranteed to be paired with either a subsequent call to `e_focus( FALSE )` or, if the edit control has been closed, a call to `e_destroy`. Deactivation events are always followed by activation events, and vice versa, until the edit control has been closed.

### Implementation Notes

It is not possible to change the focus (with `MakeFront`) in response to an `e_focus` call.

---

## XVT\_Editable::SelectText

---

### SELECT TEXT

---

#### Prototypes

```
void
SelectText(
    long                first,
    long                last )
```

#### Parameters

**first**  
The first character in the new selection. Characters are indexed from zero.

**last**  
The last character in the new selection.

#### Description

Modifies the current selection in an edit field.

If first and last are identical, the insertion point is changed.

#### Equivalent C Function

```
win_select_item_text()
```

## Implementation Members

XVT\_Editable

## Inherited Member Functions

#### From XVT\_Label

*page 239*    `void GetTitle( char* str, unsigned long* len )`

*page 239*    `virtual BOOLEAN Init( XVT_Rct boundary, long = 0L, char *  
                          = NULL )`

*page 240*    `void SetTitle( char* str )`

**From XVT\_Control**

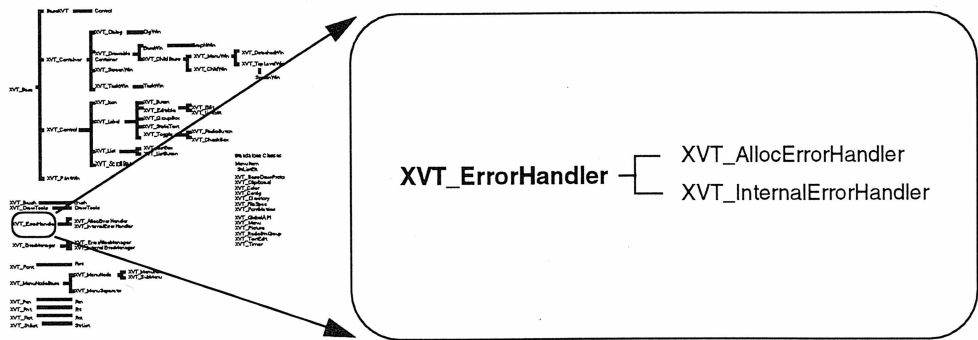
*page 92*    virtual void Close()  
*page 93*    virtual void e\_create()  
*page 93*    virtual void e\_destroy()  
*page 94*    virtual long e\_user( long id, void \*data )  
*page 95*    BOOLEAN GetEnabledState()  
*page 95*    long GetID( void )  
*page 95*    XVT\_Base \*GetParent( void )  
*page 96*    BOOLEAN GetVisibleState()  
*page 96*    void Init()  
*page 96*    void MakeFront()  
*page 97*    void SetEnabledState( BOOLEAN state )  
*page 98*    void SetInnerRect( XVT\_Rct boundary )  
*page 98*    void SetVisibleState( BOOLEAN state )

**From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()

*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_ErrorHandler



## Overview

Header File	error.h
Source File	error.cc
Superclass	
Subclasses	XVT_AllocErrorHandler, XVT_InternalErrorHandler
Usage	Abstract

This class defines the interface to an error handler. Error handlers are used in combination with error managers, which manage the handling of types of errors. For any type of error there is a subclass of XVT\_ErrorManager and a subclass of XVT\_ErrorHandler. Chains of error handlers are maintained by error managers and invoked using the error manager's Raise member function.

## Constructors

```
XVT_ErrorHandler( XVT_ErrorManager *manager )
    Create an error handler for the type of errors being handled by
    the given error manager. The newly created handler is pushed
    on the chain of handlers maintained by the error manager.

virtual ~XVT_ErrorHandler()
```

## Member Functions

---

### XVT\_ErrorHandler::Handle

---

HANDLE AN ERROR

---

#### Prototypes

```
virtual BOOLEAN
Handle(
    long                                data ) = 0
```

#### Parameters

data  
The data associated with the error.

#### Return Value

TRUE if the handler resolved the error condition and program execution can continue and FALSE if the next handler in the chain should be tried.

#### Description

By convention, subclasses provide a virtual function named `Handler` whose arguments correspond to those of `Raise` from the corresponding error manager. `Handle` is overridden in the subclass to unpack data into the original arguments passed to `Raise` and pass those to `Handler`, which will actually decide what to do.

It is possible to create `Handler` implementations that call `longjmp`; however, before doing that, you should be aware that error handlers can be invoked from the bottom of arbitrarily deep recursion involving both your application code and the window system code. The chances of completing a jump and finding the window system

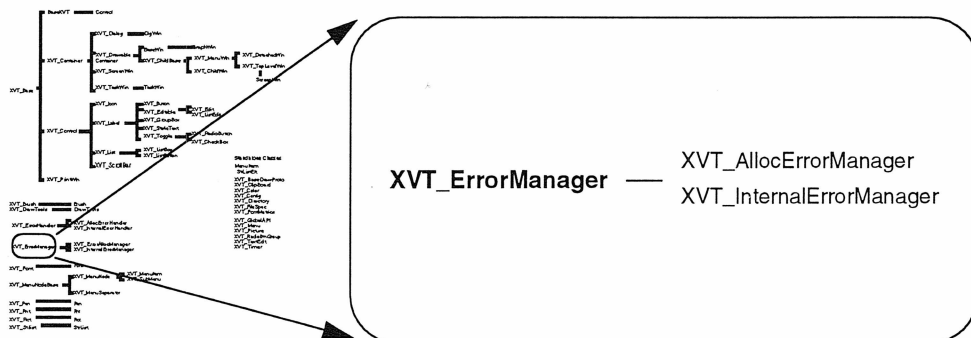
in operable condition are very poor. We suggest that you avoid `setjmp/longjmp` and instead just call `exit` from the handler directly after doing whatever cleanup you desire. That is the paradigm used by XVT++ handlers.

## **Implementation Members**

Manager



# XVT\_ErrorManager



## Overview

<b>Header File</b>	<code>error.h</code>
<b>Source File</b>	<code>error.cc</code>
<b>Superclass</b>	
<b>Subclasses</b>	<code>XVT_AllocErrorManager</code> , <code>XVT_InternalErrorManager</code>
<b>Usage</b>	Concrete

This class defines the interface common to all error managers.

An error manager manages the handling of a particular type of error, for example, memory allocation errors. Each new type of error requires an error manager subclass customized to handle it.

# Constructors

```
XVT_ErrorManager()  
~XVT_ErrorManager()
```

## Member Functions

---

### XVT\_ErrorManager::Raise

RAISE AN ERROR

---

#### Prototypes

```
void  
Raise(  
    long  
    data )
```

#### Parameters

`data`  
The data associated with this error.

#### Return Value

If `Raise` returns, the user can assume that the error condition has been handled and that the operation that caused the condition may be retried.

#### Description

Notifies the error manager that an error, described by `data`, has occurred. Usually, this function is not used directly. Instead, subclasses implement two `raise` functions, one with identical parameters to this one and one with a convenient set of parameters for the programmer (for example, a string, `__FILE__`, `__LINE__`). The latter function packs up its arguments and invokes the former, which in turn just calls this function.

## Implementation Members

`PushHandler`  
`RemoveHandler`  
`Chain`



## Operators

XVT\_FileSpec& operator=( const XVT\_FileSpec& file\_spec )

## Member Functions

---

### XVT\_FileSpec::GetDir

RETRIEVE THE DIRECTORY

---

#### Prototypes

XVT\_Directory  
GetDir() const

#### Return Value

The directory part of a file specification.

---

### XVT\_FileSpec::GetName

RETRIEVE THE FILE NAME

---

#### Prototypes

BOOLEAN  
GetName(  
    char\*                    buffer,  
    unsigned long\*          len ) const

#### Parameters

buffer  
    Storage to receive the file name.  
len  
    A pointer to the length of buffer.

#### Return Value

TRUE if the length of buffer was sufficient to hold the file name,  
FALSE if not. If FALSE is returned, len is set to the required length.

---

## XVT\_FileSpec::GetType

RETRIEVE THE FILE TYPE

---

### Prototypes

```
BOOLEAN  
GetType(  
    char*          buffer,  
    unsigned long* len ) const
```

### Parameters

**buffer**  
Storage to receive the type name.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the type name,  
FALSE if not. If FALSE is returned, len is set to the required length.

---

## XVT\_FileSpec::SetDir

SET THE DIRECTORY

---

### Prototypes

```
void  
SetDir(  
    XVT_Directory    d )
```

### Parameters

**d**  
The new directory part of the file specification.

### Description

Sets the directory part of a file specification.

---

## XVT\_FileSpec::SetName

SET THE FILE NAME

---

### Prototypes

```
void  
SetName(  
    const char*      str )
```

### Parameters

str  
The new file name.

### Description

Sets the file name.

---

## XVT\_FileSpec::SetType

SET THE FILE TYPE

---

### Prototypes

```
void  
SetType(  
    const char*      type )
```

### Parameters

type  
The new file type.

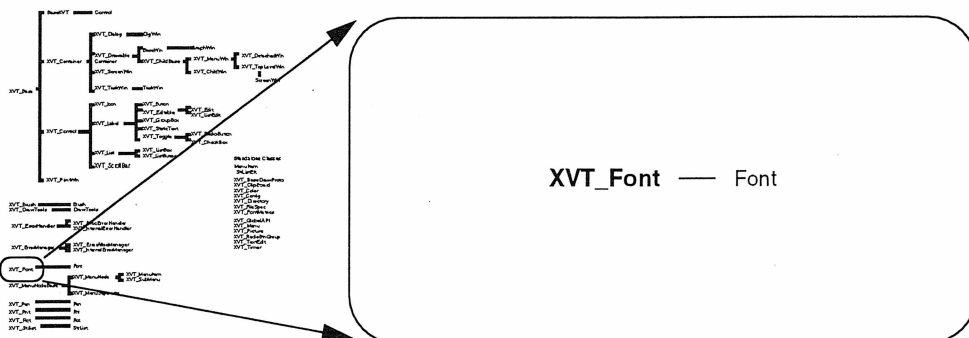
### Description

Sets the file type.

## Implementation Members

```
ConvertTo  
Dir  
Type  
Name
```

# XVT Font



## Overview

<b>Header File</b>	tools.h
<b>Source File</b>	tools.cc
<b>Superclass</b>	
<b>Subclasses</b>	Font
<b>Usage</b>	Concrete

Instances of the `XVT_Font` class specify particular fonts. The font object is entirely opaque. You cannot portably modify the internal components of a font once it has been instantiated.

The only legitimate ways to obtain an instance of this class are via an `e_font` member function, by calling `GetDrawTools`, or by constructing the font based on family style and size parameters.

## Constructors

```
XVT_Font()  
XVT_Font( long family, long style, short size )  
    Create a font that best matches the given family, style and size.  
    Equivalent to the C function select_font.  
XVT_Font( const XVT_Font& font )  
~XVT_Font()
```

## Operators

```
XVT_Font& operator=( const XVT_Font& font )  
    Fonts may be assigned.
```

## Member Functions

---

### XVT\_Font::GetSize

RETRIEVE THE FONT'S SIZE

---

#### Prototypes

```
short  
GetSize() const
```

#### Return Value

The font's size.

---

### XVT\_Font::SetSize

SET THE FONT'S SIZE

---

#### Prototypes

```
void  
SetSize(  
    short size )
```



### Parameters

size  
The new font size.

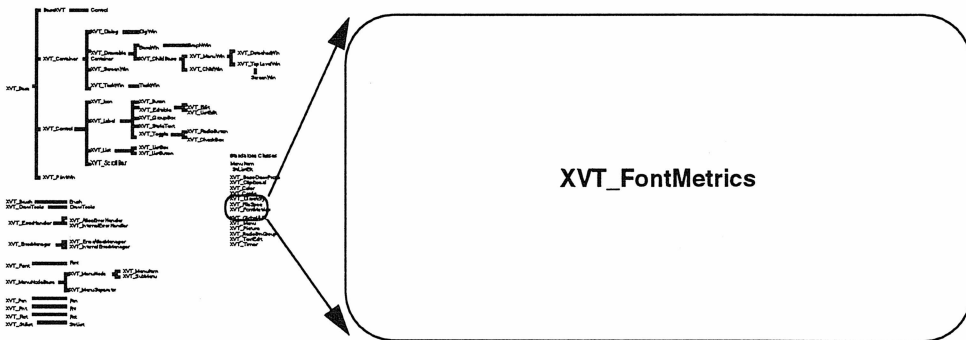
### Description

Set the font's size.

## Implementation Members

GetStyle  
SetStyle  
GetFamily  
SetFamily  
ConvertTo  
ConvertFrom  
Style  
Family  
Size

# XVT\_FontMetrics



# Overview

<b>Header File</b>	tools.h
<b>Source File</b>	tools.cc
<b>Superclass</b>	
<b>Subclasses</b>	
<b>Usage</b>	Concrete

Instances of the `XVT_FontMetrics` class describe a font in terms of its leading, ascent, and descent. Instances of this class are used to provide data to the application's text setting computations.

Usually, instances of this class are obtained from the member function `XVT_BaseDrawProto::GetFontMetrics`.

To single-space a font, augment the y coordinate of each successive line of text by the sum of the ascent, descent, and leading.

## Constructors

```
XVT_FontMetrics(  
    long leading = 0,  
    long ascent = 0,  
    long descent = 0 )  
XVT_FontMetrics( const XVT_FontMetrics& Metrics )  
~XVT_FontMetrics
```

## Operators

```
XVT_FontMetrics& operator=(  
    const XVT_FontMetrics& metrics )  
BOOLEAN operator==( const XVT_FontMetrics& metrics )
```

## Member Functions

---

### XVT\_FontMetrics::GetAscent

RETRIEVE THE ASCENT

---

#### Prototypes

```
long  
GetAscent() const
```

#### Return Value

The ascent.

---

### XVT\_FontMetrics::GetDescent

RETRIEVE THE DESCENT

---

#### Prototypes

```
long  
GetDescent() const
```

#### Return Value

The descent.

---

## XVT\_FontMetrics::GetLeading

RETRIEVE THE LEADING

---

### Prototypes

```
long  
GetLeading() const
```

### Return Value

The leading.

---

## XVT\_FontMetrics::SetAscent

SET THE ASCENT

---

### Prototypes

```
void  
SetAscent(  
    long                ascent )
```

### Parameters

```
ascent  
    The new ascent.
```

### Description

Sets the ascent.

The ascent is the distance from the baseline to the top of the tallest character in a font.

---

## XVT\_FontMetrics::SetDescent

SET THE DESCENT

---

### Prototypes

```
void  
SetDescent(  
    long                descent )
```

**Parameters**

descent  
The new descent.

**Description**

Sets the descent.  
The descent is the distance from the baseline to the bottom of the lowest character.

---

## XVT\_FontMetrics::SetLeading

SET THE LEADING

---

**Prototypes**

```
void  
SetLeading(  
    long                l )
```

**Parameters**

l  
The new leading.

**Description**

Sets the leading.  
The leading is the distance between the baselines of adjacent (single-spaced) lines of text minus the ascent and descent.

## Implementation Members

Leading  
Ascent  
Descent



## Constructors

```
XVT_GlobalAPI()
~XVT_GlobalAPI()
```

## Member Functions

---

### XVT\_GlobalAPI::About

DISPLAY AN ABOUT DIALOG

---

#### Prototypes

```
void
About()
```

#### Description

Displays an about dialog as specified by the `XVT_Config` structure given to the task window. XVT provides a standard about box resource in `url.h`. You may override the standard about box by using `XVT_Config::SetAboutBoxID` to your own about box resource ID.

If you choose to specify your own about dialog resource, it should contain buttons with control IDs of `DLG_CANCEL` and `DLG_OK` in addition to whatever static text or icons you desire. If the user presses `DLG_OK`, About calls Help. If the user presses `DLG_CANCEL`, About dismisses the about dialog.

#### Equivalent C Function

```
about_box()
```

---

### XVT\_GlobalAPI::Ask

ASK THE USER A QUESTION

---

#### Prototypes

```
ASK_RESPONSE
Ask(
    const char*    lbl_dflt,
    const char*    lbl2,
    const char*    lbl3,
    const char*    fmt... )
```

**Parameters****lbl\_dflt**

The title for the default button.

**lbl2**

The title for the second response button. Passing a NULL value causes the second response button not to be displayed.

**lbl3**

The title for the third response button. Passing a NULL value causes the third response button not to be displayed.

**fmt**The question, a `printf` style format string and arguments. The total length of the formatted question must be less than 200 characters.**Return Value****RESP\_DEFAULT**The user clicked on the button whose title was given by `lbl_dflt`.**RESP\_2**The user clicked on the button whose title was given by `lbl2`.**RESP\_3**The user clicked on the button whose title was given by `lbl3`.**Description**

Puts up a dialog that asks the user a question and offers two or three possible responses. For a dialog with just one response, use `Note` or `Error`.

**Equivalent C Function**`xvt_ask()`

---

**XVT\_GlobalAPI::Beep**PRODUCE AN AUDIBLE BEEP

---

**Prototypes**

```
void
Beep()
```

**Description**

Makes a standard beep sound. Usually used to indicate an error.



**Equivalent C Function**`xvt_beep()`

---

**XVT\_GlobalAPI::ChgDir**CHANGE THE CURRENT DIRECTORY

---

**Prototypes**

```
void
ChgDir(
    XVT_Directory    dir )
```

**Parameters**

`dir`  
The new current directory.

**Description**

Changes the current directory.

**Equivalent C Function**`chg_dir()`

---

**XVT\_GlobalAPI::Debug**APPEND DEBUG INFORMATION TO A FILE

---

**Prototypes**

```
void
Debug(
    const char*      fmt ... )
```

**Parameters**

`fmt`  
An sprintf-style format and arguments that yield the debug message. If a NULL is passed, the debug file will be closed and re-opened.

**Description**

Appends a debug message to a file. The file is named **DEBUG** and it appears in the directory that was current when the first call to Debug was made.

**Equivalent C Function**`xvt_dbg()`

---

**XVT\_GlobalAPI::Debug2**

---

CONDITIONALLY APPEND DEBUG INFORMATION TO A FILE

---

**Prototypes**

```
void
Debug2(      const char*      fmt ... )
```

**Parameters**

`fmt`  
An `sprintf`-style format and arguments that yield the debug message. If a `NULL` is passed, the debug file will be closed and re-opened.

**Description**

This function behaves identically to `Debug` if the preprocessor symbol `DEBUG` is defined when the file containing this call is compiled and a file named **XVTDEBUG** is present in the startup directory. If either of those conditions is not met, this function does nothing.

**Equivalent C Function**`dbg2()`

---

**XVT\_GlobalAPI::Error**

---

DISPLAY AN ALERT BOX WITH AN ERROR ICON

---

**Prototypes**

```
void
Error(      const char*      fmt... )
```

**Parameters**

fmt

An sprintf-style format and arguments that give the error message. The total length of the formatted message must be less than 200 characters.

**Description**

Puts up an alert box containing an error message, an error icon, and an OK button. When the user presses OK, the dialog completes and Error returns. This dialog should be used only to indicate recoverable errors on the part of the user. Application errors should be communicated using Fatal or Message.

**Equivalent C Function**

xvt\_error()

---

## XVT\_GlobalAPI::Fatal

DISPLAY AN ALERT BOX AND TERMINATE

---

**Prototypes**

```
void  
Fatal(  
    const char*          fmt... )
```

**Parameters**

fmt

An sprintf-style format and arguments that give the error message. The total length of the formatted message must be less than 200 characters.

**Description**

Puts up an alert box containing an error message, an error icon, and an OK button. When the user presses OK, the dialog completes and the application is terminated. The error message is also written to a file called **DEBUG** in case attempting to display the dialog causes a crash.

**Equivalent C Function**

xvt\_fatal()

---

## XVT\_GlobalAPI::FindEOL

---

BREAK AN ARRAY OF CHARACTERS INTO INDIVIDUAL LINES

---

### Prototypes

```
char*
FindEOL(
    const char*    buffer,
    long           nbytes,
    long*          len,
    EOL_FORMAT*    fp )
```

### Parameters

**buffer**  
The string to scan for end-of-lines or NULL to indicate that the scan is to continue from where it left off.

**nbytes**  
The number of bytes in buffer.

**len**  
The length of the returned line.

**fp**  
The type of line-end sequence. If the value returned is EOL\_NORMAL, it indicates that the line was terminated, using the same termination sequence as the first line. A value of EOL\_DIFF indicates that different line termination sequences have been detected, and a value of EOL\_NONE indicates that the final line was not terminated by an EOL sequence.

### Return Value

A pointer to the start of the line, or NULL if no lines remain.

### Description

Breaks an array of characters into individual lines by searching for native end-of-line sequences. Initially this function should be called with a string argument for **buffer**. Subsequent calls should pass a NULL value for **buffer** indicating that the value provided in the initial call should be used. **FindEOL** will continue to return lines until it reaches the end of **buffer** at which point it will return NULL.

### Equivalent C Function

`find_eol()`

---

## XVT\_GlobalAPI::GAlloc

ALLOCATE A GLOBAL MEMORY BLOCK

---

### Prototypes

```
GHANDLE
GAlloc( long          size )
```

### Parameters

**size**  
The size in bytes of the block of memory to allocate.

### Return Value

A valid GHANDLE if successful, (GHANDLE)0 if not.

### Description

This function allocates memory from the “global” heap. The global heap is a separate memory manager that has special characteristics that vary between platforms. You may consider using global memory to reduce heap fragmentation on the Mac and Windows platforms.

This function returns a GHANDLE representing the memory allocated. A GHANDLE is *not* a pointer. To get a pointer to the memory, you call GLock and pass it to GHANDLE. When you are not using the pointer, you call GUnlock to allow the system to possibly move the memory block and defragment the heap.

Once a global memory block is allocated, you can get its size with GSize, resize it with GReAlloc, or free it with GFree.

You must not assume that the portable use of XVT global memory supports any of the tricks available on the Mac or Windows. In particular, global memory is *not* shared memory! Do not attempt to pass GHANDLES from one application to another, any more than you would pass a pointer from one application to another.

### Implementation Notes

#### XVT/Mac

If you are planning to run your application on the Mac, then you can use GAlloc to allocate memory that can be moved by the Mac operating system to another location. Doing so avoids heap fragmentation, and allows your application to use less memory. Typically, the memory saved is on the order of 20%. Of course,

the trade-off is that your application requires more complexity to manage the locking and unlocking required to use global memory, and will suffer a performance hit due to the locking and unlocking overhead.

#### XVT/Win

If you are planning to run your application on Windows, then using `GAlloc` will allocate memory segments that can be moved by the operating system. However, there are three problems with this approach. First, the number of memory blocks that can be allocated via `GAlloc` is limited to about 2000. Second, each memory block carries at least 30 bytes of overhead. Third, the performance of global memory is poor (typically 40 times slower than an average heap manager). Therefore, on Windows, there is no way to get the heap-defragmentation benefits of movable memory without these penalties.

If you do not use global memory, and instead allow the heap to be fragmented, then additional memory will be used. This, in turn, will manifest itself as some additional disk access, since Windows uses virtual memory. In our opinion, the additional disk access is still faster than using global memory.

#### Equivalent C Function

`galloc()`

---

## XVT\_GlobalAPI::GetAttrValue

RETRIEVE A VALUE FROM THE SYSTEM ATTRIBUTE TABLE

---

#### Prototypes

```
long
GetAttrValue(
    XVT_Base*      win,
    long           attribute )
```

#### Parameters

**win**  
The window whose attribute is to be modified, or NULL if no window is applicable.

**attribute**  
The attribute code.

**Return Value**

The value of the given attribute.

**Equivalent C Function**

get\_value()

---

## XVT\_GlobalAPI::GetDefaultBackColor

RETRIEVE THE DEFAULT BACKGROUND COLOR

---

**Prototypes**

XVT\_Color  
GetDefaultBackColor()

**Return Value**

The systemwide default background color. This function returns an XVT\_Color object that can be used directly in calls to the drawing functions. See the description of the ATTR\_BACK\_COLOR attribute on page 210.

**Equivalent C Function**

get\_front\_top\_level\_window()

---

## XVT\_GlobalAPI::GetFrontTopLevelWin

RETRIEVE THE FRONTMOST TOP-LEVEL WINDOW

---

**Prototypes**

XVT\_MenuWin\*  
GetFrontTopLevelWin()

**Return Value**

The frontmost top-level window. A top-level window is one whose parent is either the task window or the screen window.

**Equivalent C Function**

get\_front\_top\_level\_window()

---

## XVT\_GlobalAPI::GetFrontWin

GET THE FRONTMOST WINDOW

---

### Prototypes

```
XVT_ChildBase*
GetFrontWin()
```

### Return Value

The frontmost window with keyboard focus.

### Equivalent C Function

```
get_front_window()
```

---

## XVT\_GlobalAPI::GetDefaultDir

RETRIEVE THE DEFAULT DIRECTORY

---

### Prototypes

```
XVT_Directory
GetDefaultDir()
```

### Return Value

The default directory. The default directory is the conceptual representation of the current directory, equivalent to “.” in UNIX, DOS, and OS/2 systems. It is distinct from the value returned by `GetDir`, which is simply a particular directory that happens to be current at that time.

### Equivalent C Function

```
get_default_dir()
```



---

## XVT\_GlobalAPI::GetDialogUserData

RETRIEVE USER DATA ASSOCIATED WITH A CONTROL IN A DIALOG

---

### Prototypes

```
BOOLEAN  
GetDialogUserData(  
    char*          buffer,  
    long           rid,  
    long           cid,  
    long           data_tag,  
    unsigned long* len )
```

### Parameters

**buffer**  
The buffer for the user data string.

**rid**  
The resource ID of the dialog.

**cid**  
The control-ID of the control. If cid is 0, the user data for the dialog is returned.

**data\_tag**  
The index of the user data. Indexes start at 0 and increase.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the user data, FALSE if not. If FALSE is returned, len is set to the required length. If len is 0, no such user data exists.

### Description

Retrieves user data associated with a control in a dialog or with a dialog.

### Equivalent C Function

get\_dialog\_userdata()

---

## XVT\_GlobalAPI::GetDir

RETRIEVE THE CURRENT DIRECTORY

---

### Prototypes

```
XVT_Directory
GetDir()
```

### Return Value

The current directory.

### Equivalent C Function

```
get_dir()
```

---

## XVT\_GlobalAPI::GetMenuUserData

RETRIEVE USER DATA ASSOCIATED WITH A MENU ITEM

---

### Prototypes

```
BOOLEAN
GetMenuUserData(
    char*          buffer,
    long           rid,
    MENU_TAG       menu_tag,
    long           data_tag,
    unsigned long* len )
```

### Parameters

**buffer**  
The buffer for the user data string.

**rid**  
The resource ID of the menu.

**menu\_tag**  
The tag of the menu item.

**data\_tag**  
The index of the user data. Indexes start at 0 and increase.

**len**  
A pointer to the length of buffer.

**Return Value**

TRUE if the length of buffer was sufficient to hold the user data, FALSE if not. If FALSE is returned, len is set to the required length. If len is 0, no such user data exists.

**Description**

Retrieves user data associated with a menu item.

**Equivalent C Function**

get\_menu\_userdata()

---

## XVT\_GlobalAPI::GetResString

RETRIEVE A STRING FROM RESOURCES

---

**Prototypes**

```
BOOLEAN  
GetResString(  
    char*  
    long  
    unsigned long*  
    buffer,  
    rid,  
    len )
```

**Parameters**

buffer  
The buffer for the resource string.

rid  
The resource ID of the string.

len  
A pointer to the length of buffer.

**Return Value**

TRUE if the length of buffer was sufficient to hold the string, FALSE if not. If FALSE is returned, len is set to the required length.

**Description**

Retrieves a string from resources.

**Equivalent C Function**

get\_res\_str()

---

## XVT\_GlobalAPI::GetWindowUserData

---

RETRIEVE USER DATA ASSOCIATED WITH A CONTROL IN A WINDOW

---

### Prototypes

```
BOOLEAN
GetWindowUserData(
    char*          buffer,
    long           rid,
    long           cid,
    long           data_tag,
    unsigned long* len )
```

### Parameters

**buffer**  
The buffer for the user data string.

**rid**  
The resource ID of the window.

**cid**  
The control-ID of the control. If cid is 0, the user data for the window is returned.

**data\_tag**  
The index of the user data. Indexes start at 0 and increase.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the user data, FALSE if not. If FALSE is returned, len is set to the required length. If len is 0, no such user data exists.

### Description

Retrieves user data associated with a control in a window or with a window.

### Equivalent C Function

get\_window\_userdata()

---

## XVT\_GlobalAPI::GFree

FREE A BLOCK OF GLOBAL MEMORY

---

### Prototypes

```
void  
GFree(  
    GHANDLE  
        handle )
```

### Parameters

handle  
The handle to the block to be freed.

### Description

Frees a block of global memory.

### Equivalent C Function

gfree()

---

## XVT\_GlobalAPI::GLock

LOCK DOWN A GLOBAL BLOCK OF MEMORY

---

### Prototypes

```
char*  
GLock(  
    GHANDLE  
        handle )
```

### Parameters

handle  
The handle to the block to be locked down.

### Return Value

A pointer to the memory block itself.

### Description

Locks down a global block of memory. The memory will not be relocated by the system until the block is unlocked.

### Equivalent C Function

glock()

---

## XVT\_GlobalAPI::GReAlloc

RESIZE A GLOBAL BLOCK OF MEMORY

---

### Prototypes

```
GHANDLE  
GReAlloc(  
    GHANDLE    handle,  
    long        size )
```

### Parameters

**handle**  
The handle to be resized.

**size**  
The new size of the block.

### Return Value

A valid GHANDLE if successful, (GHANDLE)0 if not.

### Description

Resizes a global block of memory to have size size. As the address of the block may change, the block should be unlocked before calling this function.

### Equivalent C Function

grealloc()

---

## XVT\_GlobalAPI::GSize

RETRIEVE THE SIZE OF A GLOBAL BLOCK

---

### Prototypes

```
long  
GSize(  
    GHANDLE    handle )
```

### Parameters

**handle**  
A handle to a block of global memory.

### Return Value

The size of the global block of memory.

**Equivalent C Function**`gsize()`

---

**XVT\_GlobalAPI::GUnLock**UNLOCK A BLOCK OF MEMORY

---

**Prototypes**

```
void
GUnLock(
    GHANDLE          handle )
```

**Parameters**

`handle`  
The handle of the block to be unlocked.

**Description**

Unlocks a block of memory.

**Equivalent C Function**`gunlock()`

---

**XVT\_GlobalAPI::Help**ENTER THE HELP SYSTEM

---

**Prototypes**

```
void
Help()
```

**Description**

Enters the help system. XVT++'s help system consists of a modeless "Topics" dialog box, as well as a variable number of "Help Text" dialogs that XVT++ creates and destroys as the user browses the help system. Calling `Help` creates the "Topics" dialog. Since the "Topics" dialog is modeless, `Help` returns immediately after displaying the dialog. This allows the user to access the help system and the rest of the application simultaneously.

When `Help` is called, XVT++ first tries to locate a help file that was compiled with **CHELP**, whose name is determined by the base

application name from the XVT\_Config instance passed to the task windows Init. This file is sought in the startup directory. If XVT can't find the file, it prompts the user to find it by calling OpenFile. If the user cannot find the help file, then the help system aborts. Once the file is opened, the "Topics" dialog is displayed. The user can respond to the dialog box by choosing a topic or cancelling it. If the user chooses to cancel, they leave the help system. If the user chooses a topic, the text of the topic is displayed in a "Help Text" dialog box.

There are two cases where XVT++ will automatically call Help without the intervention of your application. The first case is when the user presses the OK button in your application's About box. The second case where XVT++ will automatically call Help is when the user chooses from your menubar a menu item with tag equal to M\_HELP.

### Equivalent C Function

```
xvt_help()
```

---

## XVT\_GlobalAPI::ListFaces

LIST AVAILABLE TYPEFACES

---

### Prototypes

```
void  
ListFaces(  
    XVT_StrList*      list )
```

### Parameters

list  
The string list to which the available typefaces are to be added.

### Description

Lists available typefaces.

### Equivalent C Function

```
list_faces()
```



---

## XVT\_GlobalAPI::ListResStrings

RETRIEVE STRINGS WITH CONSECUTIVE RESOURCEIDS

---

### Prototypes

```
void  
ListResStrings(  
    XVT_StrList*    dest,  
    long            rid_first,  
    long            rid_last )
```

### Parameters

**dest**  
The string list to which the strings will be added.

**rid\_first**  
The resource ID of the first string.

**rid\_last**  
The resource ID of the last string.

### Description

Retrieves strings with consecutive resource IDs from the resource file.

### Equivalent C Function

```
list_res_str()
```

---

## XVT\_GlobalAPI::Message

DISPLAY AN EMERGENCY MESSAGE

---

### Prototypes

```
void  
Message(  
    const char*    fmt... )
```

### Parameters

**fmt**  
An sprintf-style format and arguments that give the error message. The total length of the formatted message must be less than 200 characters.

**Description**

Puts up an alert box containing an error message, an error icon, and an OK button. When the user presses OK, the dialog completes and Message returns. The dialog put up does not come from resources and should not cause any extra memory or resources to be allocated so it can be successfully displayed in out-of-memory conditions.

**Equivalent C Function**

xvt\_msg()

---

**XVT\_GlobalAPI::Note**

DISPLAY AN ALERT BOX WITH A NOTE ICON

---

**Prototypes**

```
void  
Note(  
    const char*          fmt... )
```

**Parameters**

fmt  
An sprintf-style format and arguments that give the error message. The total length of the formatted message must be less than 200 characters.

**Description**

Puts up an alert box containing a message, a note icon, and an OK button. When the user presses OK, the dialog completes and Note returns.

**Equivalent C Function**

xvt\_note()

---

## XVT\_GlobalAPI::OpenFile

GET A FILE TO READ WITH A STANDARD DIALOG

---

### Prototypes

```
FL_STATUS  
OpenFile(  
    XVT_FileSpec*      file_spec,  
    const char*         str )
```

### Parameters

**file\_spec**  
The file specification to open. Set the type to be the type of files the user is allowed to select; "" for any type. Set the directory to be the directory initially presented to the user.

**str**  
A message to be displayed to the user in the dialog, "Select drawing file...", for example.

### Return Value

**FL\_OK**  
The user clicked on the OK button and selected a file. The file specification pointed to by **file\_spec** is now valid.

**FL\_BAD**  
An error occurred. An alert has already been displayed by the dialog.

**FL\_CANCEL**  
The user canceled the dialog.

### Description

Puts up a dialog box that requests the user to select a file to be opened for reading. The file is not opened; only the file specification is returned. Upon return your application must change to the proper directory, check that the file exists and is readable and finally, open it.

### Implementation Notes

**XVT/Mac**  
The file type portion of the file specification is ignored. Users can select any type of file.

### Equivalent C Function

`open_file_dlg()`

---

## XVT\_GlobalAPI::PageSetup

DISPLAY THE STANDARD PAGE SETUP DIALOG

---

### Prototypes

```
BOOLEAN  
PageSetup(  
    char*                print_record )
```

### Parameters

print\_record  
The print record.

### Return Value

A flag that is TRUE if the given print record was modified, FALSE if not.

### Description

Puts up a dialog box allowing the user to adjust the page setup stored in the given print record. It should be called in response to the user's choosing page setup on the file menu. If your application has just read the print record from a file, you should first call `ValidatePrintRcd` to make sure that the record is valid.

### Implementation Notes

XVT/CH, XVT/XOL, XVT/XM  
This function is not implemented and always returns FALSE without ever displaying a dialog.

### Equivalent C Function

`page_setup_dlg()`

---

## XVT\_GlobalAPI::ProcessEvent

PROCESS PENDING EVENTS

---

### Prototypes

```
void  
ProcessEvent()
```

**Description**

This function causes XVT++ to empty the event queue of all pending events and to dispatch them to the appropriate event handler method functions. After all events have been dispatched and the functions that received them have returned, `ProcessEvents` returns.

If you call `ProcessEvents`, you might receive a recursive call to an event handling member function. You should plan carefully for this by, among other things, restricting the use of global variables. In particular, make sure that the recursive call won't end up calling `ProcessEvents` again.

Calling `ProcessEvents` during an otherwise unbroken operation (such as loading a file), allows user input to be processed.

Therefore, call this function often (every 1/10th second suffices) during long operations, such as reading or writing a file, or when performing a time-consuming computation such as sorting. During that operation you might put up a dialog box that offers the user the opportunity to Cancel. You must call `ProcessEvents` for the dialog to function.

**Implementation Notes**

XVT/Mac, XVT/Win, XVT/PM

Calling `ProcessEvents` gives other applications a chance to execute.

**Equivalent C Function**

`process_events()`

---

## XVT\_GlobalAPI::ReadAccess

CHECK TO SEE IF FILE IS READABLE

---

**Prototypes**

```

BOOLEAN
ReadAccess(
    const char*      path )

```

**Parameters**

`path`  
The file's pathname.

**Return Value**

A flag which is TRUE if the file is readable, FALSE if it is not.

---

## XVT\_GlobalAPI::Response

Obtain a string from the user

---

### Prototypes

```
char*
Response(
    char*      prompt,
    char*      response,
    unsigned long resp_len )
```

### Parameters

prompt

The prompt to display. Only about 100 characters of prompt message can be displayed by the dialog.

response

The response buffer. On entry the value in this buffer will be used as the default response - it will be loaded into the text entry field and selected when the response dialog comes up. On exit it will contain whatever the user entered into the text entry field.

resp\_len

The length of the response buffer in bytes.

### Return Value

A pointer to the response buffer if the user entered a response, NULL if the dialog was cancelled.

### Description

Obtain a character string from the user by bringing up a modal dialog which displays a prompt and allows the user to enter a response or cancel the dialog.

### Equivalent C Function

```
get_str_response()
```

---

## XVT\_GlobalAPI::RestoreDir

RESTORE THE CURRENT DIRECTORY

---

### Prototypes

```
void
RestoreDir()
```

**Description**

Restores (changes directory to) the directory saved by the last call to SaveDir().

**Equivalent C Function**

restore\_dir()

---

## XVT\_GlobalAPI::SaveDir

SAVE THE CURRENT DIRECTORY

---

**Prototypes**

```
void
SaveDir()
```

**Description**

Saves the current directory. This call causes the previously saved directory to be forgotten.

**Equivalent C Function**

save\_dir()

---

## XVT\_GlobalAPI::SaveFile

GET A FILE TO WRITE WITH A STANDARD DIALOG

---

**Prototypes**

```
FL_STATUS
SaveFile(
    XVT_FileSpec*    file_spec,
    const char*      str )
```

**Parameters**

file\_spec

The file specification to open. Set the type to be the type of files the user is allowed to select; "" for any type. Set the directory to be the directory initially presented to the user.

str

A message to be displayed to the user in the dialog, "Select drawing file...", for example.

**Return Value**

FL\_OK

The user clicked on the OK button and selected a file. The file specification pointed to by file\_spec is now valid.

FL\_BAD

An error occurred. An alert has already been displayed by the dialog.

FL\_CANCEL

The user canceled the dialog.

**Description**

Puts up a dialog box that requests the user to select a file to be opened for writing. The file is not opened; only the file specification is returned. Upon return your application must change to the proper directory, and open the file. If the file exists, your application should prompt the user before overwriting it.

**Implementation Notes**

XVT/Mac

The file type portion of the file specification is ignored. Users can select any type of file.

**Equivalent C Function**

```
save_file_dlg()
```

---

## XVT\_GlobalAPI::SetAttrValue

SET AN ATTRIBUTE VALUE

---

**Prototypes**

```
void
SetAttrValue(
    XVT_Base*    win,
    long         attribute,
    long         value )
```

**Parameters**

win

The object whose attribute is to be modified, or NULL if no object is applicable.



attribute

The attribute code. Attribute codes are given by ATTR constants.

value

The new value for the attribute.

## Description

Modifies an entry in the system attribute table.

The ATTR\_\* constants consist of two types of values: values that are defined to be portable across all window systems, and values that are defined to be specific for a particular platform. These constants are used as the attr argument for SetAttrValue and GetAttrValue. In this section, only the attributes that are portable across all platforms are described. For a detailed description of the platform-specific attributes, refer to the platform-specific books.

### ATTR\_CH\_\*

Description:

XVT/CH platform-specific attributes.

See also:

*XVT/CH* platform-specific book

### ATTR\_MAC\_\*

Description:

XVT/Mac platform-specific attributes.

See also:

*XVT/Mac* platform-specific book

### ATTR\_PM\_\*

Description:

XVT/PM platform-specific attributes.

See also:

*XVT/PM* platform-specific book

### ATTR\_WIN\_\*

Description:

XVT/Win platform-specific attributes.

See also:

*XVT/Win* platform-specific book

## ATTR\_WIN\_PM\_\*

## Description:

Platform-specific attributes that are common to XVT/Win and XVT/PM.

## See also:

*XVT/Win* and *XVT/PM* platform-specific books

## ATTR\_XM\_\*

## Description:

XVT/XM platform-specific attributes.

## See also:

*XVT/XM* platform-specific book

## ATTR\_XOL\_\*

## Description:

XVT/XOL platform-specific attributes.

## See also:

*XVT/XOL* platform-specific book

## ATTR\_X\_\*

## Description:

Platform-specific attributes common to XVT/XM and XVT/XOL.

## See also:

*XVT/XM* and *XVT/XOL* platform-specific books

## ATTR\_BACK\_COLOR

## Description:

The system-wide window background COLOR as set by the user. Applications wishing to honor the user's settings can retrieve this color and use it in their calls to `Clear`. Be sure not to confuse this with the XVT++ drawing tools background color.

## Uses win argument:

no

## GetValue returns:

the user's choice of window background color

## SetValue effect:

invalid

## See also:

`Clear`

**ATTR\_CTL\_BUTTON\_HEIGHT****Description:**

The best-looking button height, in pixels. This value should be used to create button controls that look optimal. The optimal button width depends on the width of its label, which can be measured by calling `GetTextWidth` with the system font.

**Uses win argument:**

no

**GetValue returns:**

button height

**SetValue effect:**

invalid

**ATTR\_CTL\_CHECK\_BOX\_HEIGHT****Description:**

The best-looking check box height, in pixels. This value should be used to create check box controls that look optimal. The optimal check box width depends on the width of its label, which can be measured by calling `GetTextWidth` with the system font.

**Uses win argument:**

no

**GetValue returns:**

check box height

**SetValue effect:**

invalid

**ATTR\_CTL\_EDIT\_TEXT\_HEIGHT****Description:**

The best-looking edit control height, in pixels. This value should be used to create edit controls that look optimal.

**Uses win argument:**

no

**GetValue returns:**

edit control height

**SetValue effect:**

invalid

## ATTR\_CTL\_HORZ\_SBAR\_HEIGHT

## Description:

The best-looking horizontal scrollbar thickness, in pixels. This value is the same as the thickness of horizontal scrollbars that are created by specifying WSF\_HSCROLL when creating a window.

## Uses win argument:

no

## GetValue returns:

scrollbar thickness

## SetValue effect:

invalid

## ATTR\_CTL\_RADIOBUTTON\_HEIGHT

## Description:

The best-looking radio button height, in pixels. This value should be used to create radio button controls that look optimal. The optimal radio button width depends on the width of its label, which can be measured by calling `GetTextWidth` with the system font.

## Uses win argument:

no

## GetValue returns:

radio button height

## SetValue effect:

invalid

## ATTR\_CTL\_STATIC\_TEXT\_HEIGHT

## Description:

The best-looking static text control height, in pixels. This value should be used to create static text controls that look optimal.

## Uses win argument:

no

## GetValue returns:

check box height

## SetValue effect:

invalid

**ATTR\_CTL\_VERT\_SBAR\_WIDTH****Description:**

The best-looking vertical scrollbar thickness, in pixels. This value is the same as the thickness of vertical scrollbars that are created by specifying `WSF_VSCROLL` when creating a window.

**Uses win argument:**

no

**GetValue returns:**

scrollbar thickness

**SetValue effect:**

invalid

**ATTR\_DBLFRAME\_HEIGHT****Description:**

The thickness in pixels of a horizontal border of a double-border window. This can be used to calculate what the outer size of a window will be given its client area.

**Uses win argument:**

no

**GetValue returns:**

border thickness in pixels

**SetValue effect:**

invalid

**ATTR\_DBLFRAME\_WIDTH****Description:**

The thickness in pixels of a vertical border of a double-border window. This can be used to calculate what the outer size of a window will be given its client area.

**Uses win argument:**

no

**GetValue returns:**

border thickness in pixels

**SetValue effect:**

invalid

**ATTR\_DEBUG\_FILENAME****Description:**

The name of the debugging output file used by XVT++.

Uses win argument:

no

GetValue returns:

a pointer to a static buffer containing the current debug filename, which is "DEBUG" by default

SetValue effect:

Passing a pointer to a string containing the new debug filename causes the next debug file open to open the newly installed filename.

#### ATTR\_DOCFRAME\_HEIGHT

Description:

The thickness in pixels of a horizontal border of a resizable window. This can be used to calculate what the outer size of a window will be given its client area.

Uses win argument:

no

GetValue returns:

border thickness in pixels

SetValue effect:

invalid

#### ATTR\_DOCFRAME\_WIDTH

Description:

The thickness in pixels of a vertical border of a resizable window. This can be used to calculate what the outer size of a window will be given its client area.

Uses win argument:

no

GetValue returns:

border thickness in pixels

SetValue effect:

invalid

#### ATTR\_DOC\_STAGGER\_HORZ

Description:

Recommended horizontal document window cascading offset.

Uses win argument:

no

GetValue returns:  
offset in pixels

SetValue effect:  
invalid

#### ATTR\_DOC\_STAGGER\_VERT

Description:  
Recommended vertical document window cascading offset.

Uses win argument:  
no

GetValue returns:  
offset in pixels

SetValue effect:  
invalid

#### ATTR\_EVENT\_HOOK

Description:  
A pointer to an event-handling function for native events. The prototype of this function varies between platforms, as do the nature of events sent to it. However, all event hook functions set with this attribute have the same return value. Namely, they return TRUE if XVT++ should perform its normal processing of the native event, and FALSE if XVT++ should not process the event.

Uses win argument:  
no

GetValue returns:  
the currently installed event hook function

SetValue effect:  
Sets the event hook function. Setting this to NULL is valid, and means that there is no event hook installed.

See also:  
platform-specific books

#### ATTR\_FATAL\_ERR\_HANDLER

Description:  
A pointer to an error-handling function that will be called at the very beginning of the Fatal function's processing. This is to allow your application to perform fatal-error-specific cleanup in one place. This is especially useful for fatal errors that are

generated internally to XVT++. This function must return. The fatal error handler function prototype is:

```
void (*FATAL_ERR_FUNC)();
```

Uses win argument:

no

GetValue returns:

current fatal error handler pointer

SetValue effect:

Sets fatal error handler function pointer. Setting to NULL means that there is no error handler.

#### ATTR\_FRAME\_HEIGHT

Description:

The thickness in pixels of a horizontal border of a non-resizable window. This can be used to calculate what the outer size of a window will be given its client area.

Uses win argument:

no

GetValue returns:

border thickness in pixels

SetValue effect:

invalid

#### ATTR\_FRAME\_WIDTH

Description:

The thickness in pixels of a vertical border of a non-resizable window. This can be used to calculate what the outer size of a window will be given its client area.

Uses win argument:

no

GetValue returns:

border thickness in pixels

SetValue effect:

invalid

#### ATTR\_HAVE\_COLOR

Description:

A BOOLEAN value indicating if the program is running on a color system.



Uses win argument:

no

GetValue returns:

TRUE if the system is color

SetValue effect:

invalid

#### ATTR\_HAVE\_MOUSE

Description:

A BOOLEAN value indicating if the program is running on a system with a mouse or other pointing device present.

Uses win argument:

no

GetValue returns:

TRUE if the system has a pointing device

SetValue effect:

invalid

#### ATTR\_ICON\_HEIGHT

Description:

The default icon height. This can be used to determine how much space will be used by DrawIcon. However, it is possible to create variable-size icons on some platforms, so this value has limited usefulness.

Uses win argument:

no

GetValue returns:

icon height

SetValue effect:

invalid

#### ATTR\_ICON\_WIDTH

Description:

The default icon width. This can be used to determine how much space will be used by DrawIcon. However, it is possible to create variable-size icons on some platforms, so this value has limited usefulness.

Uses win argument:

no

GetValue returns:

icon width

SetValue effect:

invalid

#### ATTR\_KEY\_HOOK

Description:

A pointer to an event-handling function for native keystroke events. The prototype of this function varies between platforms, as do the nature of events sent to it. However, all key hook functions set with this attribute have the same return value. Namely, they return FALSE if XVT++ should perform its normal key translation, and TRUE if XVT++ should accept the key translation performed by the hook function. This allows the application to supplement XVT++'s internal key translation algorithm.

Uses win argument:

no

GetValue returns:

the currently installed key hook function

SetValue effect:

Sets the key hook function. Setting this to NULL is valid, and means that there is no key hook installed.

See also:

platform-specific books

#### ATTR\_MALLOC\_ERR\_HANDLER

Description:

A pointer to an error-handling function that is called when the XVT++ memory allocation functions `xvt_malloc` and `xvt_realloc` run out of memory. The function has the following prototype:

```
BOOLEAN (*MEM_ERR_FUNC)(size_t size);
```

Where `size` is the amount of memory needed. If you install a malloc error handler, then it should return TRUE if it is somehow able to make more memory available (such as by freeing a pre-allocated block), or FALSE otherwise.

Uses win argument:

no

GetValue returns:

current malloc error handler pointer

**SetValue effect:**

Sets malloc error handler function pointer. Setting to NULL means that there is no error handler.

**See also:**

The “Memory Allocation” chapter in the *XVT Guide*, for use of this attribute.

**ATTR\_MENU\_HEIGHT****Description:**

The height of a menubar. This can be used to calculate what the outer size of a window will be given its client area. However, it is up to the application to determine whether a particular window has a menu attached to it.

**Uses win argument:**

no

**GetValue returns:**

menu height in pixels

**SetValue effect:**

invalid

**ATTR\_NATIVE\_GRAPHIC\_CONTEXT****Description:**

This value represents the underlying graphical context used by the native window system, for a particular window. While this is a “portable” attribute, it has a non-portable return value. For Windows, this returns an HDC. For PM, this returns an HPS. For Mac, this returns a Grafport. For X platforms (XM and XOL), this returns a GC. However, we do not recommend using this GC, as it has undocumented side-effects, and GCs are easy to create yourself.

**Uses win argument:**

yes

**GetValue returns:**

native context (requires casting)

**SetValue effect:**

invalid

**See also:**

platform-specific books

## ATTR\_NATIVE\_WINDOW

## Description:

This value represents the underlying window object used by the native window system, for a particular window. While this is a “portable” attribute, it has a non-portable return value. For Windows and PM, this returns an HWND. For Mac, this returns a Windowptr. For XM and XOL, this returns a Window.

## Uses win argument:

yes

## GetValue returns:

native graphical window (requires casting)

## SetValue effect:

invalid

## See also:

platform-specific books

## ATTR\_NUM\_TIMERS

## Description:

The number of timers in the system available to the application via XVT\_Timer objects.

## Uses win argument:

no

## GetValue returns:

number of available timers

## SetValue effect:

invalid

## ATTR\_PRINTER\_HEIGHT

## Description:

The height of the default printer, in pixels.

## Uses win argument:

no

## GetValue returns:

printer height

## SetValue effect:

invalid

## See also:

XVT\_ESC\_GET\_PRINTER\_INFO in the *XVT/Mac* and *XVT/PM* platform-specific books

## ATTR\_PRINTER\_HRES

## Description:

The horizontal resolution of the default printer, in pixels per inch.

## Uses win argument:

no

## GetValue returns:

printer horizontal resolution

## SetValue effect:

invalid

## See also:

XVT\_ESC\_GET\_PRINTER\_INFO in the *XVT/Mac* and *XVT/PM* platform-specific books

## ATTR\_PRINTER\_WIDTH

## Description:

The width of the default printer, in pixels.

## Uses win argument:

no

## GetValue returns:

printer width

## SetValue effect:

invalid

## See also:

XVT\_ESC\_GET\_PRINTER\_INFO in the *XVT/Mac* and *XVT/PM* platform-specific books

## ATTR\_PRINTER\_VRES

## Description:

The vertical resolution of the default printer, in pixels per inch.

## Uses win argument:

no

## GetValue returns:

printer vertical resolution

## SetValue effect:

invalid

## See also:

XVT\_ESC\_GET\_PRINTER\_INFO in the *XVT/Mac* and *XVT/PM* platform-specific books

**Note:** ATTR\_PRINTER\_\* only return values appropriate for the default printer settings. To retrieve printer metrics for a printer setting stored in a PRINT\_RCD, see the non-portable XVT\_ESC\_GET\_PRINTER\_INFO found in the platform-specific books.

#### ATTR\_SCREEN\_HEIGHT

Description:  
The height of the screen, in pixels.

Uses win argument:  
no

GetValue returns:  
screen height

SetValue effect:  
invalid

#### ATTR\_SCREEN\_HRES

Description:  
The horizontal resolution of the screen, in pixels per inch.

Uses win argument:  
no

GetValue returns:  
screen horizontal resolution

SetValue effect:  
invalid

#### ATTR\_SCREEN\_VRES

Description:  
The vertical resolution of the screen, in pixels per inch.

Uses win argument:  
no

GetValue returns:  
screen vertical resolution

SetValue effect:  
invalid

#### ATTR\_SCREEN\_WIDTH

Description:  
The width of the screen, in pixels.

Uses win argument:  
no

GetValue returns:  
screen width

SetValue effect:  
invalid

See also:  
*The Guide*

#### ATTR\_SUPPRESS\_UPDATE\_CHECK

Description:

A BOOLEAN value that controls XVT++'s policing of invalid function calls during calls to `e_update`. Normally, XVT++ disallows many function calls during an `e_update`, because they confuse the native window systems and are poor programming practice. However, if your application runs into an obscure case requiring this check to be disabled, then you can set this attribute to TRUE.

Uses win argument:  
no

GetValue returns:  
TRUE if update checking is disabled

SetValue effect:  
Disables update checking if TRUE. Enables update checking if FALSE.

#### ATTR\_TITLE\_HEIGHT

Description:

The height of a window's title. This can be used to calculate what the outer size of a window will be given its client area. However, it is up to the application to determine whether a particular window has a title attached to it.

Uses win argument:  
no

GetValue returns:  
menu height in pixels

SetValue effect:  
invalid

#### Equivalent C Function

`set_value()`

---

## XVT\_GlobalAPI::SetFileType

SET A FILE'S TYPE AND CREATOR

---

### Prototypes

```
void  
SetFileType(  
    XVT_FileSpec*    file_spec,  
    const char*      creator )
```

### Parameters

**file\_spec**  
The file whose type and creator are to be set.

**creator**  
A null terminated string specifying the creator. This string should be no longer than 4 alphanumeric characters.

### Description

Set a file's type and creator if these are specified separately from a file's name.

### Implementation Notes

XVT/Mac  
This is the only platform on which this function is not a no-op. On all other platforms, the file type is just part of the file name, and the concept of creator doesn't exist.

---

## XVT\_GlobalAPI::StartupDir

RETURN TO THE APPLICATION'S STARTUP DIRECTORY

---

### Prototypes

```
void  
StartupDir()
```

### Description

Returns to the application's startup directory.

### Equivalent C Function

```
startup_dir()
```



---

## XVT\_GlobalAPI::TranslatePoints

TRANSLATE POINTS RELATIVE TO CONTAINERS

---

### Prototypes

```
void  
TranslatePoints(  
    XVT_Container*    from,  
    XVT_Container*    to,  
    XVT_Pnt*          points,  
    long              count )
```

### Parameters

from

The container to whose coordinate system the points are currently relative.

to

The container whose coordinate system the points should be translated into.

points

The array of points to be translated.

count

The number of points in points.

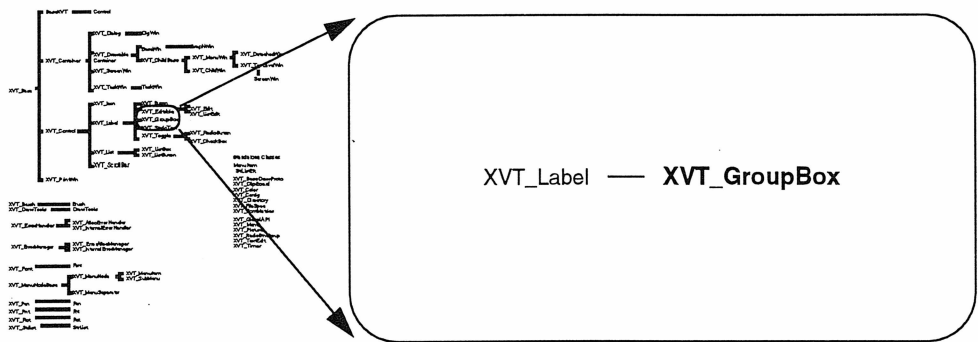
### Description

Translates points from one container coordinate system to another.

### Equivalent C Function

translate\_points()

# XVT\_GroupBox



## Overview

Header File	groupbox.h
Source File	
Superclass	XVT_Label
Subclasses	
Usage	Concrete

This class defines the interface to group box controls. As there are no virtual event handling member functions you do not need to subclass XVT\_GroupBox to get a working control; just instantiate the class directly.

XVT++ group box controls provide a way to draw an annotated rectangle around (and behind) a group of controls in a window or dialog. The group box rectangle has an embedded label or title, which appears on the upper line of the rectangle, and may be either left, centered, or right depending on the text justification flags for the control.

A group box defines those controls that are within its boundaries as a set. This does not imply that a group box is the parent of controls contained within it; no such relationship exists. Group boxes are like

static text in that they provide no interaction capability or subsequent events; they are for annotation purposes only.

Group boxes are automatically placed to the back of a dialog or window by XVT++, behind all other controls. The behavior of overlapping group boxes is undefined.

## Constructors

XVT\_GroupBox( XVT\_Dialog\* parent, long cid )

XVT\_GroupBox( XVT\_DrawableContainer\* parent, long cid )

## Inherited Member Functions

### From XVT\_Label

page 239 void GetTitle( char\* str, unsigned long\* len )

page 239 virtual BOOLEAN Init( XVT\_Rct boundary, long = 0L, char \*  
= NULL )

page 240 void SetTitle( char\* str )

### From XVT\_Control

page 92 virtual void Close()

page 93 virtual void e\_create()

page 93 virtual void e\_destroy()

page 94 virtual long e\_user( long id, void \*data )

page 95 BOOLEAN GetEnabledState()

page 95 long GetID( void )

page 95 XVT\_Base \*GetParent( void )

page 96 BOOLEAN GetVisibleState()

page 96 void Init()

page 96 void MakeFront()

page 97 void SetEnabledState( BOOLEAN state )

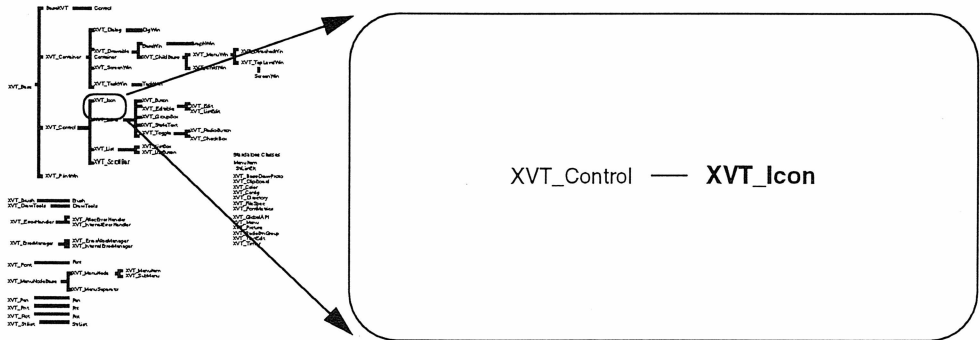
page 98 void SetInnerRect( XVT\_Rct boundary )

page 98 void SetVisibleState( BOOLEAN state )

**From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_Icon



## Overview

<b>Header File</b>	icon.h
<b>Source File</b>	icon.cc
<b>Superclass</b>	XVT_Control
<b>Subclasses</b>	
<b>Usage</b>	Concrete

This class defines the interface to icons. Since icons do not receive any events, there is no need to subclass XVT\_Icon to produce a working icon. Just instantiate it directly.

XVT++ icon controls allow you to display platform-specific icons in dialogs and windows. The actual description (or resource definition) of an icon is handled differently for each XVT++ platform (see the platform-specific books for details.) However, once icons are described, XVT++ can portably handle their inclusion into windows and dialogs.

## Constructors

```
XVT_Icon( XVT_Dialog* parent, long cid )
```

```
XVT_Icon( XVT_DrawableContainer* parent, long cid )
```

## Member Functions

---

### XVT\_Icon::Init

INITIALIZE AN ICON

---

#### Prototypes

```
virtual BOOLEAN  
Init()
```

```
virtual BOOLEAN  
Init(  
    XVT_Rct          boundary,  
    long             cid = 0L,  
    long             flags = 0L )
```

#### Parameters

**boundary**  
The extent (outer boundary) of the icon.

**cid**  
The icon's resource ID.

**flags**  
A bitwise OR'd combination of control attribute flags.

#### Return Value

TRUE if the control was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by controls. Recovery can be attempted by disposing of the new control, closing another control, and retrying the creation of the control.

#### Description

Create the native icon if it does not already exist. If the icon is in a window or dialog that was created from resources, the underlying icon will already exist and the `XVT_Control::Init` member function should be used instead.

**Equivalent C Function**

```
create_control()
create_def_control()
```

**Implementation Members**

```
virtual BOOLEAN Init( XVT_IconEntry* icon_def )
```

**Inherited Member Functions****From XVT\_Control**

```

page 92    virtual void Close()
page 93    virtual void e_create()
page 93    virtual void e_destroy()
page 94    virtual long e_user( long id, void *data )
page 95    BOOLEAN GetEnabledState()
page 95    long GetID( void )
page 95    XVT_Base *GetParent( void )
page 96    BOOLEAN GetVisibleState()
page 96    void Init()
page 96    void MakeFront()
page 97    void SetEnabledState( BOOLEAN state )
page 98    void SetInnerRect( XVT_Rct boundary )
page 98    void SetVisibleState( BOOLEAN state )
```

**From XVT\_Base**

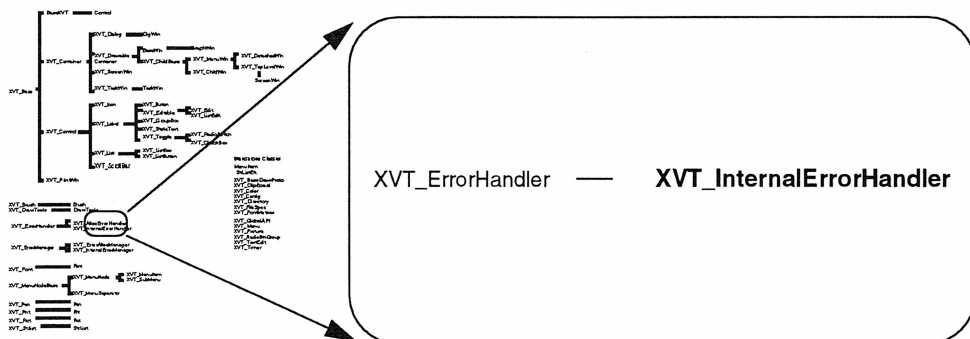
```

page 11    virtual BaseWin* CastToBaseWin()
page 10    virtual DlgWin* CastToDlgWin()
page 10    virtual ScreenWin* CastToScreenWin11()
page 10    virtual TaskWin* CastToTaskWin11()
page 11    virtual XVT_Button *CastToButton()
page 11    virtual XVT_CheckBox *CastToCheckBox()
```

*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()



# XVT\_InternalErrorHandler



## Overview

Header File	error.h
Source File	error.cc
Superclass	XVT_ErrorHandler
Subclasses	
Usage	Abstract

This class defines the interface to all internal error handlers. To create your own internal error handler, you would create a subclass that provides an implementation of `Handler`, which does whatever it needs to.

## Example

Suppose that you were working with a database that your application had to lock and unlock. Even if an internal error occurred, it would be nice if your application released any locks it was holding before

exiting so that you would not have to do this by hand. The following subclass achieves this goal:

```
class MyInternalErrorHandler : public
XVT_InternalErrorHandler
{
    BOOLEAN Handler(
        char*file,
        char*version,
        longline,
        char*msg );
}

BOOLEAN
MyInternalErrorHandler::Handler(
    char* file,
    char* version,
    long line,
    char* msg )
{
    // Release any locks here
    return FALSE;
}
```

## Constructors

XVT\_InternalErrorHandler()

## Member Functions

---

### XVT\_InternalErrorHandler::Handler

HANDLE AN INTERNAL ERROR

---

#### Prototypes

protected:

```
virtual BOOLEAN
Handler(
    const char*      file,
    const char*      version,
    long             line,
    const char*      msg )
```

## Parameters

**file**  
The file in which the error occurred.

**version**  
A string identifying the version of the file in which the error occurred.

**line**  
The line number where the error occurred.

**msg**  
A message describing the error.

## Return Value

TRUE if the handler resolved the error condition and program execution can continue, FALSE if the next handler in the chain should be tried.

Since it is not possible to recover from an internal error, `Handle` ignores the value returned from `Handler` and always returns FALSE.

## Description

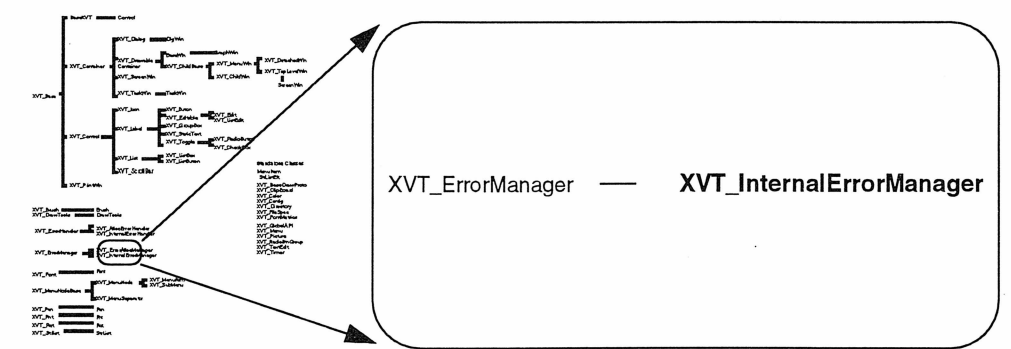
This function is called by `Handle` when this error handler is given a chance to handle an internal error.

# Inherited Member Functions

## From XVT\_ErrorHandler

*page 167*     `virtual BOOLEAN Handle( long data )`

# XVT\_InternalErrorManager



## Overview

Header File	error.h
Source File	error.cc
Superclass	XVT_ErrorManager
Subclasses	
Usage	Concrete

Instances of this class handle XVT++ internal errors. These errors arise when assertions inside XVT++ fail. They indicate a problem in the usage of XVT++.

There is only one instance of this class, pointed to by the global variable XVT\_InternalError.

## Constructors

XVT\_InternalErrorManager()

## Member Functions

---

### XVT\_InternalErrorManager::Raise

RAISE AN XVT++ INTERNAL ERROR

---

#### Prototypes

```
void
Raise(
    const char*    file,
    const char*    version,
    long           line,
    const char*    msg )
```

#### Parameters

**file**  
The file in which the error occurred.

**version**  
The RCS version of the file in which the error occurred.

**line**  
The line number where the error occurred.

**msg**  
A message describing the error.

#### Description

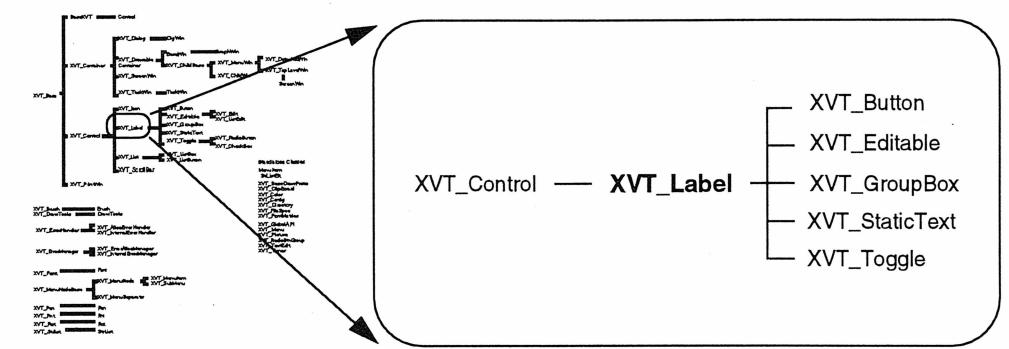
Signals an internal error. This function is always called through the macro `XVT_INTERNAL_ERROR`, which takes `msg` as a parameter and supplies the other three parameters.

## Inherited Member Functions

#### From XVT\_ErrorManager

*page 170*    virtual void Raise( long data )

# XVT\_Label



## Overview

Header File	label.h
Source File	label.cc
Superclass	XVT_Control
Subclasses	XVT_Button, XVT_Editable, XVT_GroupBox, XVT_StaticText, XVT_Toggle
Usage	Implementation

The XVT\_Label class defines the interface common to all controls that have settable titles.

## Member Functions

---

### XVT\_Label::GetTitle

RETRIEVE A CONTROL'S TITLE

---

#### Prototypes

```

    BOOLEAN
    GetTitle(
        char*          buffer,
        unsigned long* len ) const
  
```

#### Parameters

**buffer**  
Storage to receive the control's title.

**len**  
A pointer to the length of buffer.

#### Return Value

TRUE if the length of **buffer** was sufficient to hold the application's name, FALSE if not. If FALSE is returned, **len** is set to the required length.

#### Equivalent C Function

```
get_title()
```

---

### XVT\_Label::Init

INITIALIZE A LABEL

---

#### Prototypes

```

    BOOLEAN
    Init()

    BOOLEAN
    Init(
        XVT_Rct    boundary,
        long        flags = 0L,
        const char* title = NULL )
  
```

**Parameters**

boundary

The bounding rectangle for the control. If the height of the bounding rectangle is zero, the default height of the native system is used.

flags

Attribute flags.

title

The control's initial title.

**Return Value**

TRUE if the control was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by controls. Recovery may be attempted by disposing of the new control, closing another control, and retrying the creation of the control.

**Description**

Creates the native control if it does not already exist. If the control is in a window or dialog that was created from resources, the underlying control already exists and the XVT\_Control::Init member function should be used instead.

**Equivalent C Function**

create\_control()

create\_def\_control()

---

**XVT\_Label::SetTitle**

SETA CONTROL'S TITLE

---

**Prototypes**

```
void
SetTitle(
    const char*      str )
```

**Parameters**

str

The new title.

**Description**

Sets the control's title to the title passed in str.



**Equivalent C Function**

```
set_title()
```

**Implementation Members**

```
virtual BOOLEAN Init( XVT_ControlEntry* def )
TitleProtocol
```

**Inherited Member Functions****From XVT\_Control**

```

page 92    virtual void Close()
page 93    virtual void e_create()
page 93    virtual void e_destroy()
page 94    virtual long e_user( long id, void *data )
page 95    BOOLEAN GetEnabledState()
page 95    long GetID( void )
page 95    XVT_Base *GetParent( void )
page 96    BOOLEAN GetVisibleState()
page 96    void Init()
page 96    void MakeFront()
page 97    void SetEnabledState( BOOLEAN state )
page 98    void SetInnerRect( XVT_Rct boundary )
page 98    void SetVisibleState( BOOLEAN state )
```

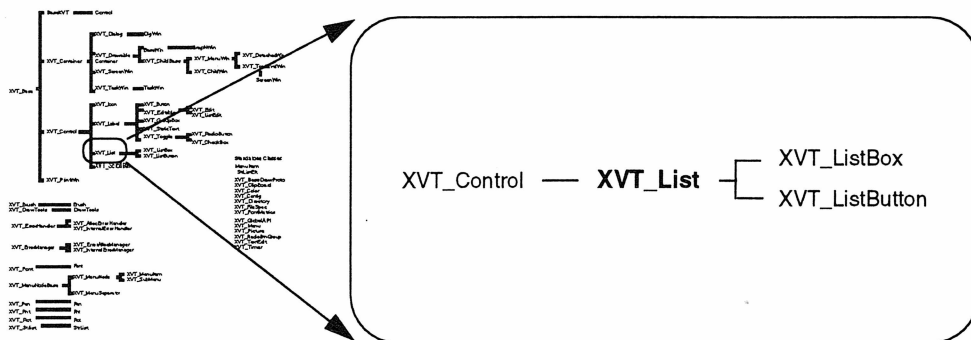
**From XVT\_Base**

```

page 11    virtual BaseWin* CastToBaseWin()
page 10    virtual DlgWin* CastToDlgWin()
page 10    virtual ScreenWin* CastToScreenWin11()
page 10    virtual TaskWin* CastToTaskWin11()
page 11    virtual XVT_Button *CastToButton()
page 11    virtual XVT_CheckBox *CastToCheckBox()
```

<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer *CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>

# XVT\_List



## Overview

<b>Header File</b>	list.h
<b>Source File</b>	list.cc
<b>Superclass</b>	XVT_Control
<b>Subclasses</b>	XVT_ListBox, XVT_ListButton
<b>Usage</b>	Implementation

The `XVT_List` class defines the interface common to all objects that have list components.

## Member Functions

---

### XVT\_List::Add

ADD AN ITEM OR ITEMS TO A LIST

---

#### Prototypes

```

BOOLEAN
Add(
    long                index,
    const char*         str )

BOOLEAN
Add(
    const char*         str )

BOOLEAN
Add(
    long                index,
    XVT_StrList*        list )

BOOLEAN
Add(
    XVT_StrList*        list )

```

#### Parameters

**index**  
The index of the item before which to add the new item or items. An index that is too large or -1 causes items to be added to the end of the list.

**str**  
The text of the item to add.

**list**  
The list of items to add.

#### Description

```

Add( index, str )
    Add a string to the list control at the location given by index.

Add( str )
    Add a string to the end of the list control.

Add( index, list )
    Add a list of strings to the list control at the location given by index.

Add( list )
    Add a list of strings to the end of the list control.

```

**Equivalent C Function**`win_list_add`

---

**XVT\_List::Clear**

REMOVE ALL ITEMS FROM THE LIST

---

**Prototypes**`BOOLEAN  
Clear()`**Return Value**

TRUE if successful, FALSE if not.

**Description**

Removes all items from the list.

**Equivalent C Function**`win_list_clear()`

---

**XVT\_List::CountAll**

RETRIEVE THE NUMBER OF ITEMS IN A LIST

---

**Prototypes**`long  
CountAll() const`**Return Value**

The number of items in the list.

**Equivalent C Function**`win_list_count_all()`

---

## XVT\_List::CountSelections

RETRIEVE THE NUMBER OF SELECTED ITEMS

---

### Prototypes

```
long  
CountSelections()
```

### Return Value

The number of selected items in the list. For single select list boxes this is always either 1 or 0.

### Equivalent C Function

```
win_list_count_sel()
```

---

## XVT\_List::Delete

REMOVE AN ITEM FROM A LIST CONTROL

---

### Prototypes

```
BOOLEAN  
Delete(  
    long  
        index )
```

### Parameters

index  
The index of the item to delete.

### Return Value

TRUE if successful, FALSE if not.

### Description

Deletes an item from a list control.

### Equivalent C Function

```
win_list_delete()
```

---

## XVT\_List::GetAll

RETRIEVE ALL ITEMS FROM A LIST CONTROL

---

### Prototypes

```
XVT_StrList  
GetAll() const
```

### Return Value

A list of all items in the list control.

### Equivalent C Function

```
win_list_get_all()
```

---

## XVT\_List::GetElement

RETRIEVE AN ITEM IN A LIST CONTROL

---

### Prototypes

```
BOOLEAN  
GetElement(  
    long          index,  
    char*         buffer,  
    unsigned long* len )
```

### XVT\_List::Parameters

**index**  
The index of the item to get.

**buffer**  
Storage to receive the item.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the selected item,  
FALSE if not. If FALSE is returned, len is set to the required length.

### Equivalent C Function

```
win_list_get_elt()
```

---

## XVT\_List::GetFirstSelection

RETRIEVE THE FIRST SELECTED ITEM IN A LIST BOX

---

### Prototypes

```
BOOLEAN  
GetFirstSelection(  
    char*                buffer  
    unsigned long*       len ) const
```

### Parameters

**buffer**  
Storage to receive the selected item. If no items were selected, the empty string, "", will be returned. Since empty strings can be inserted into list boxes, you should always use CountSelections to determine if there are selected items.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the selected item, FALSE if not. If FALSE is returned, len is set to the required length.

### Equivalent C Function

```
win_list_get_first_sel()
```

---

## XVT\_List::GetSelectedState

DETERMINE IF AN ITEM IS SELECTED

---

### Prototypes

```
BOOLEAN  
GetSelectedState(  
    long                index ) const
```

### Parameters

**index**  
The index of the item to check for selectedness.

### Return Value

A flag that is TRUE if the item is selected, FALSE if unselected.



**Equivalent C Function**`win_list_is_sel()`

---

**XVT\_List::GetSelectionIndex**

---

RETRIEVE THE INDEX OF THE FIRST SELECTED ITEM

---

**Prototypes**`long  
GetSelectionIndex() const`**Return Value**

The index of the first selected item in the control.

**Equivalent C Function**`win_list_get_sel_index()`

---

**XVT\_List::GetSelections**

---

RETRIEVE ALL SELECTED ITEMS

---

**Prototypes**`XVT_StrList  
GetSelections() const`**Return Value**

A string list of all selected items. The order of items in the list is the same as the order of items in the control. The data word in the list is an index to the corresponding item in the control.

**Equivalent C Function**`win_list_get_sel()`

---

## XVT\_List::Init

INITIALIZE A LIST

---

### Prototypes

```
virtual BOOLEAN
Init(
    XVT_Rct          boundary,
    long             flags = 0L,
    const char*      title = NULL )
```

### Parameters

**boundary**  
The bounding rectangle for the control. If the height of the rectangle is zero, the default height of the native system is used.

**flags**  
Attribute flags

**title**  
The list's title.

### Return Value

TRUE if the control was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by controls. Recovery may be attempted by disposing of the new control, closing another control, and retrying the creation of the control.

### Description

Create the native control if it does not already exist. If the control is in a window or dialog that was created from resources, the underlying control already exists and the `XVT_Control::Init` member function should be used instead.

---

## XVT\_List::SetSelectedState

SELECT OR UNSELECT AN ITEM

---

### Prototypes

```
void
SetSelectedState(
    long             index,
    BOOLEAN          select )
```

**Parameters**

index

The index of the item to check for selectedness.

select

A flag that is TRUE if the item is to be selected, FALSE if unselected.

**Description**

Selects or unselects an item.

**Equivalent C Function**

win\_list\_set\_sel()

**Implementation Members**

XVT\_List

~XVT\_List

virtual BOOLEAN Init( XVT\_ControlEntry\* def )

ListEltProtocol

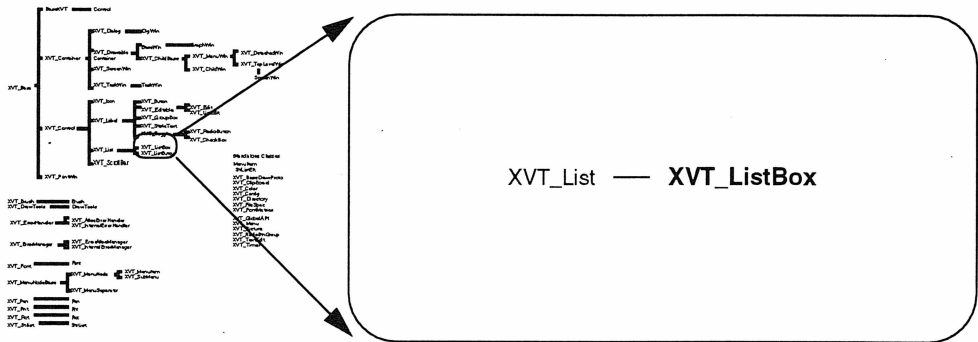
**Inherited Member Functions****From XVT\_Control**

*page 92*    virtual void Close()  
*page 93*    virtual void e\_create()  
*page 93*    virtual void e\_destroy()  
*page 94*    virtual long e\_user( long id, void \*data )  
*page 95*    BOOLEAN GetEnabledState()  
*page 95*    long GetID( void )  
*page 95*    XVT\_Base \*GetParent( void )  
*page 96*    BOOLEAN GetVisibleState()  
*page 96*    void Init()  
*page 96*    void MakeFront()  
*page 97*    void SetEnabledState( BOOLEAN state )  
*page 98*    void SetInnerRect( XVT\_Rct boundary )  
*page 98*    void SetVisibleState( BOOLEAN state )

**From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_ListBox



## Overview

<b>Header File</b>	listbox.h
<b>Source File</b>	listbox.cc
<b>Superclass</b>	XVT_List
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The XVT\_ListBox class specifies the interface to list boxes.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

List boxes allow the user to make single or multiple selections from a scrollable list of candidate selections. List boxes generate calls to the `e_action` member function when the user single clicks or double clicks on an item in the list box. You will not receive any events in your application when the user scrolls the list box; this behavior is handled automatically by the native list box control.

## Constructors

```
XVT_ListBox( XVT_Dialog *parent, long cid )  
XVT_ListBox( XVT_DrawableContainer *parent, long cid )  
virtual ~XVT_ListBox()
```

## Member Functions

---

### XVT\_ListBox::e\_action

RECEIVE NOTIFICATION OF LIST BOX ACTIVITY

---

#### Prototypes

```
virtual void  
e_action(  
    BOOLEAN                dbl_click )
```

#### Parameters

**dbl\_click**  
A flag that is TRUE if the user double-clicked on a particular item.

#### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to user manipulations of a list box.

---

### XVT\_ListBox::GetSuspendedState

DETERMINE IF UPDATES TO A LIST BOX ARE SUSPENDED

---

#### Prototypes

```
BOOLEAN  
GetSuspendedState() const
```

#### Return Value

A flag that is TRUE if updates have been suspended, FALSE if not.

---

## XVT\_ListBox::SetSuspendedState

SUSPEND OR RESUME UPDATES TO A LIST BOX

---

### Prototypes

```
void
SetSuspendedState(
    BOOLEAN                state )
```

### Parameters

**state**  
A flag that is TRUE if updates are to be suspended, FALSE if they are to be resumed.

### Description

Suspends or resumes updates to a list box.

As updating a list box can be quite costly, it is a good idea to suspend updates before a section of code that makes several modifications to a list box and resume updates after all of the modifications are completed.

### Equivalent C Function

```
win_list_suspend()
win_list_resume()
```

## Inherited Member Functions

### From XVT\_List

<i>page 244</i>	BOOLEAN Add( long index, const char* str )
<i>page 244</i>	BOOLEAN Add( const char* str )
<i>page 244</i>	BOOLEAN Add( long index XVT_StrList* list )
<i>page 244</i>	BOOLEAN Add( XVT_StrList* list )
<i>page 245</i>	BOOLEAN Clear()
<i>page 245</i>	long CountAll()
<i>page 246</i>	long CountSelections()
<i>page 246</i>	BOOLEAN Delete( long index)
<i>page 247</i>	XVT_StrList GetAll()

- page 247*    `BOOLEAN GetElement( long index, char *buffer, unsigned long* len )`
- page 248*    `BOOLEAN GetFirstSelection( char* buffer, unsigned long* len )`
- page 248*    `BOOLEAN GetSelectedState( long )`
- page 249*    `long GetSelectionIndex()`
- page 249*    `XVT_StrList GetSelections()`
- page 250*    `virtual BOOLEAN Init( XVT_Rct boundary, long flags = 0L, const char* title = NULL )`
- page 250*    `void SetSelectedState( long, BOOLEAN )`

### **From XVT\_Control**

- page 92*    `virtual void Close()`
- page 93*    `virtual void e_create()`
- page 93*    `virtual void e_destroy()`
- page 94*    `virtual long e_user( long id, void *data )`
- page 95*    `BOOLEAN GetEnabledState()`
- page 95*    `long GetID( void )`
- page 95*    `XVT_Base *GetParent( void )`
- page 96*    `BOOLEAN GetVisibleState()`
- page 96*    `void Init()`
- page 96*    `void MakeFront()`
- page 97*    `void SetEnabledState( BOOLEAN state )`
- page 98*    `void SetInnerRect( XVT_Rct boundary )`
- page 98*    `void SetVisibleState( BOOLEAN state )`

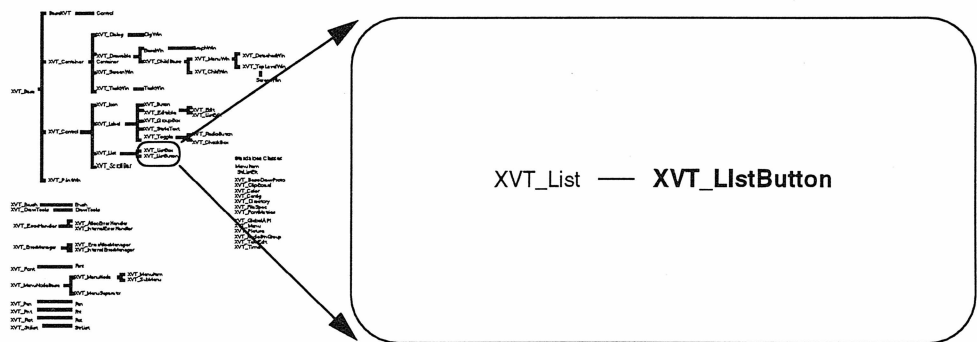
### **From XVT\_Base**

- page 11*    `virtual BaseWin* CastToBaseWin()`
- page 10*    `virtual DlgWin* CastToDlgWin()`
- page 10*    `virtual ScreenWin* CastToScreenWin11()`
- page 10*    `virtual TaskWin* CastToTaskWin11()`
- page 11*    `virtual XVT_Button *CastToButton()`



*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_ListButton



## Overview

Header File	listbtn.h
Source File	listbtn.cc
Superclass	XVT_List
Subclasses	
Usage	Abstract

The XVT\_ListButton class specifies the interface to list buttons.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

An XVT list button control is a combination of two other control types: a push button and a selection list. (Such controls are sometimes referred to as “combo controls” for this reason.) A list button can be described as a list box that can be displayed in two ways:

- A push button whose text label represents the current selection in the list (when the control is not being used).
- A list box (when the control is being used).

The list box part of the list button is transitory—it appears only when the list button is pressed. When a selection is made from the list, the list box part of the control disappears, leaving the selected text in the list button. (If the list button list is empty, then the list button label will also be empty.)

The events that are generated from list buttons are similar to those generated from list boxes except that, because double clicks aren't supported in list buttons, the event is merely signalling that the user made a selection from the list.

## Constructors

```
XVT_ListButton( XVT_Dialog* parent, long cid )
XVT_ListButton( XVT_DrawableContainer* parent, long cid )
```

## Member Functions

---

### XVT\_ListButton::e\_action

---

RECEIVE NOTICE OF USER MANIPULATION OF A LIST BUTTON

---

#### Prototypes

```
virtual void
e_action()
```

#### Description

Receives notice of user manipulation of a list button.

This member function must be overridden by a subclass if the application wishes to take any actions in response to user manipulations of a list button.

## Inherited Member Functions

#### From XVT\_List

```
page 244    BOOLEAN Add( long index, const char* str )
page 244    BOOLEAN Add( const char* str )
page 244    BOOLEAN Add( long index XVT_StrList* list )
```

<i>page 244</i>	<code>BOOLEAN Add( XVT_StrList* list )</code>
<i>page 245</i>	<code>BOOLEAN Clear()</code>
<i>page 245</i>	<code>long CountAll()</code>
<i>page 246</i>	<code>long CountSelections()</code>
<i>page 246</i>	<code>BOOLEAN Delete( long index)</code>
<i>page 247</i>	<code>XVT_StrList GetAll()</code>
<i>page 247</i>	<code>BOOLEAN GetElement( long index, char *buffer, unsigned long* len )</code>
<i>page 248</i>	<code>BOOLEAN GetFirstSelection( char* buffer, unsigned long* len )</code>
<i>page 248</i>	<code>BOOLEAN GetSelectedState( long )</code>
<i>page 249</i>	<code>long GetSelectionIndex()</code>
<i>page 249</i>	<code>XVT_StrList GetSelections()</code>
<i>page 250</i>	<code>virtual BOOLEAN Init( XVT_Rct boundary, long flags = 0L, const char* title = NULL )</code>
<i>page 250</i>	<code>void SetSelectedState( long, BOOLEAN )</code>

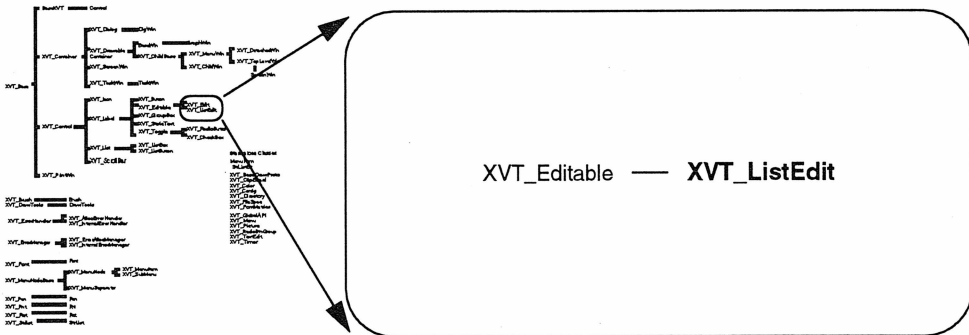
### **From XVT\_Control**

<i>page 92</i>	<code>virtual void Close()</code>
<i>page 93</i>	<code>virtual void e_create()</code>
<i>page 93</i>	<code>virtual void e_destroy()</code>
<i>page 94</i>	<code>virtual long e_user( long id, void *data )</code>
<i>page 95</i>	<code>BOOLEAN GetEnabledState()</code>
<i>page 95</i>	<code>long GetID( void )</code>
<i>page 95</i>	<code>XVT_Base *GetParent( void )</code>
<i>page 96</i>	<code>BOOLEAN GetVisibleState()</code>
<i>page 96</i>	<code>void Init()</code>
<i>page 96</i>	<code>void MakeFront()</code>
<i>page 97</i>	<code>void SetEnabledState( BOOLEAN state )</code>
<i>page 98</i>	<code>void SetInnerRect( XVT_Rct boundary )</code>
<i>page 98</i>	<code>void SetVisibleState( BOOLEAN state )</code>

**From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_ListEdit



# Overview

<b>Header File</b>	listedit.h
<b>Source File</b>	listedit.cc
<b>Superclass</b>	XVT_Editable
<b>Subclasses</b>	
<b>Usage</b>	Abstract

This class defines the interface to list edit field controls.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

# Constructors

```
XVT_ListEdit( XVT_Dialog* parent, long cid )
XVT_ListEdit( XVT_DrawableContainer* parent, long cid )
virtual ~XVT_ListEdit()
```

## Member Functions

The following functions work exactly as for XVT\_List:

<i>page 244</i>	BOOLEAN Add( long index XVT_StrList* list )
<i>page 245</i>	BOOLEAN Clear()
<i>page 245</i>	long CountAll()
<i>page 246</i>	BOOLEAN Delete( long index)
<i>page 247</i>	XVT_StrList GetAll()
<i>page 247</i>	BOOLEAN GetElement( long index, char *buffer, unsigned long* len )

The following functions work exactly as for XVT\_ListBox:

<i>page 255</i>	void SetSuspendedState( BOOLEAN state )
-----------------	---

---

## XVT\_ListEdit::Add

ADD ITEMS TO A LIST

---

### Prototypes

```
void
Add(
    long          index,
    const char*   str )

void
Add(
    const char*   str )

void
Add(
    XVT_StrList* list )
```

### XVT\_List::Parameters

**index**  
The index of the item before which to add the new item or items. An index that is too large or -1 causes items to be added at the end of the list.

**str**  
The text of the item to add.

**list**  
The list of items to add.

**Description**

Add( index, list )  
 Adds a list of items to the list control.

Add( index, str )  
 Adds a single item to the list control.

**Equivalent C Function**

win\_list\_add()

**Implementation Members**

ListEltProtocol  
 ListSuspendProtocol

**Inherited Member Functions****From XVT\_Editable**

*page 161*    virtual void e\_action()  
*page 162*    e\_focus( BOOLEANactive )  
*page 163*    void SelectText( long first, long last )

**From XVT\_Label**

*page 239*    void GetTitle( char\* str, unsigned long\* len )  
*page 239*    virtual BOOLEAN Init( XVT\_Rct boundary, long = 0L, char \*  
                              = NULL )  
*page 240*    void SetTitle( char\* str )

**From XVT\_Control**

*page 92*    virtual void Close()  
*page 93*    virtual void e\_create()  
*page 93*    virtual void e\_destroy()  
*page 94*    virtual long e\_user( long id, void \*data )  
*page 95*    BOOLEAN GetEnabledState()  
*page 95*    long GetID( void )  
*page 95*    XVT\_Base \*GetParent( void )



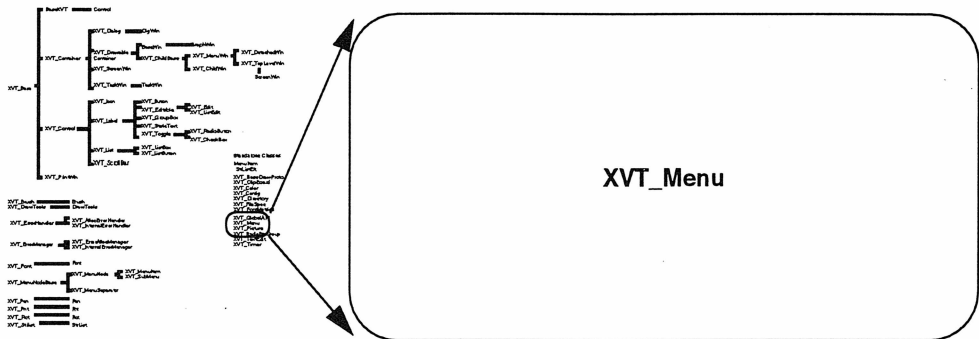
page 96     `BOOLEAN GetVisibleState()`  
page 96     `void Init()`  
page 96     `void MakeFront()`  
page 97     `void SetEnabledState( BOOLEAN state )`  
page 98     `void SetInnerRect( XVT_Rct boundary )`  
page 98     `void SetVisibleState( BOOLEAN state )`

### **From XVT\_Base**

page 11     `virtual BaseWin* CastToBaseWin()`  
page 10     `virtual DlgWin* CastToDlgWin()`  
page 10     `virtual ScreenWin* CastToScreenWin11()`  
page 10     `virtual TaskWin* CastToTaskWin11()`  
page 11     `virtual XVT_Button *CastToButton()`  
page 11     `virtual XVT_CheckBox *CastToCheckBox()`  
page 11     `virtual XVT_ChildWin *CastToChildWin()`  
page 11     `virtual XVT_DetachedWin *CastToDetachedWin()`  
page 11     `virtual XVT_Dialog *CastToDialog()`  
page 11     `virtual XVT_DrawableContainer*CastToDrawableContainer()`  
page 11     `virtual XVT_Edit *CastToEdit()`  
page 11     `virtual XVT_GroupBox *CastToGroupBox()`  
page 11     `virtual XVT_Icon *CastToIcon()`  
page 11     `virtual XVT_ListBox *CastToListBox()`  
page 11     `virtual XVT_ListButton *CastToListButton()`  
page 11     `virtual XVT_ListEdit *CastToListEdit()`  
page 11     `virtual XVT_MenuWin *CastToMenuWin()`  
page 11     `virtual XVT_PrintWin *CastToPrintWin()`  
page 11     `virtual XVT_RadioButton *CastToRadioButton()`  
page 11     `virtual XVT_ScreenWin *CastToScreenWin()`  
page 11     `virtual XVT_ScrollBar *CastToScrollBar()`  
page 11     `virtual XVT_StaticText *CastToStaticText()`

*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_Menu



## Overview

<b>Header File</b>	menu.h
<b>Source File</b>	menu.cc
<b>Superclass</b>	
<b>Subclasses</b>	
<b>Usage</b>	Concrete

A menu is a recursive structure used to specify the appearance and function of the menubar associated with a window. Conceptually, the menu consists of a list of nodes, each node specifying an item in the menu. Nodes can be separators, submenus or menu items. A submenu item points to another XVT\_Menu structure. A menu item must be subclassed so as to override its virtual event handler member function with an implementation that actually does something when the item is selected by the user.

Menus can be constructed at runtime or from a resource description. To construct a menu at runtime, build from the bottom of the menu hierarchy to the top. Create the bottom-most sub-menu, install its items, then create its parent menu, install the submenu and any other items in it and so forth until you reach the top-level menu.

To construct a menu from resources, use the `XVT_Menu( rid )` constructor. It creates a complete menu hierarchy populated with default menu items. A default menu item raises an internal error when it is selected by the user. In order to make the menu usable, you must replace the default items with instances of your own menu subclasses by using the `Replace` member function.

## Example

Let's consider loading a menu from resources as we do when we create the task window. First we create our task window subclass:

```
class MyTask : public XVT_TaskWin
{
    void e_create();
    void e_close();
    .
    .
}
```

Then we create some menu item subclasses for the standard menu items in the file menu:

```
class MyFileOpenItem : public XVT_MenuItem
{
    MyFileOpenItem( MENU_TAG tag )
        : XVT_MenuItem( tag );
    void e_action( BOOLEAN shift, BOOLEAN control );
}

void
MyFileOpenItem::e_action( BOOLEAN shift, BOOLEAN control
)
{
    // Open a file...
}

class MyFileCloseItem : public XVT_MenuItem
{
    MyFileCloseItem ( MENU_TAG tag )
        : XVT_MenuItem( tag );
}
```

```

        void e_action( BOOLEAN shift, BOOLEAN control );
    }

    void
    MyFileCloseItem::e_action(BOOLEAN shift, BOOLEAN control
    )
    {
        // Close a file...
    }

    .
    .
    .

```

Next, when the task window is created, we replace the default items, which will raise an error if they are used, with our items which do whatever we want (presumably opening and closing files) when they are used.

```

void
MyTask::e_create()
{
    XVT_MenuItem* thisItem;

    thisItem = new MyFileOpenItem( M_FILE_OPEN );
    Menu->Replace( thisItem );
    thisItem = new MyFileCloseItem( M_FILE_CLOSE );
    Menu->Replace( thisItem );
    .
    .
    .
}

```

The standard file open and close menu items will now do whatever you have programmed into the corresponding `e_action` event handler methods.

## Constructors

```

XVT_Menu()
    Create a menu at runtime. You will have to add menu items to
    the menu using the Install member function.

XVT_Menu( long rid )
    Create a menu from the given menu resource. You will need to
    replace the default menu items using the Replace member
    function.

XVT_Menu( XVT_Menu& menu )
~XVT_Menu()

```

## Member Functions

---

### XVT\_Menu::GetCount

RETRIEVE THE NUMBER OF MENU ITEMS

---

#### Prototypes

```
long  
GetCount() const
```

#### Return Value

The number of menu items in this menu.

---

### XVT\_Menu::GetFirst

RETRIEVE THE FIRST MENU ITEM

---

#### Prototypes

```
XVT_MenuNodeBase*  
GetFirst()
```

#### Return Value

The first menu item or NULL if the menu contains no items.

#### Description

Retrieves the first menu item and sets up the traversal context such that subsequent calls to `GetNext` retrieve subsequent items.

---

### XVT\_Menu::GetItem

RETRIEVE THE ITEM WITH THE MATCHING TAG

---

#### Prototypes

```
XVT_MenuNode*  
GetItem(  
    MENU_TAG          tag )
```

**Parameters**

tag  
The tag.

**Return Value**

The menu item whose tag is equal to tag or NULL if none was found.

---

## XVT\_Menu::GetNext

RETRIEVE SUBSEQUENT MENU ITEMS

---

**Prototypes**

```
XVT_MenuNodeBase*
GetNext()
```

**Return Value**

The next menu item or NULL if the end of the list of items has been reached.

**Description**

Retrieves subsequent menu items.

---

## XVT\_Menu::Install

INSTALL AN ITEM IN A MENU

---

**Prototypes**

```
void
Install(
    XVT_MenuNodeBase*    node )
```

**Parameters**

node  
The node to be installed.

**Description**

Installs a menu item in a menu. Items appear in the menu in the order in which they were installed.

This function is used to construct menus at runtime. To create menus from resources, use Replace.

---

## XVT\_Menu::Replace

REPLACE THE DEFAULT MENU ITEM WITH THE SAME TAG

---

### Prototypes

```
void  
Replace(  
    XVT_MenuItem*      item )
```

### Parameters

item  
The item to replace the default item.

### Description

Replaces the default item with the tag matching that in item. Only default items can be replaced.

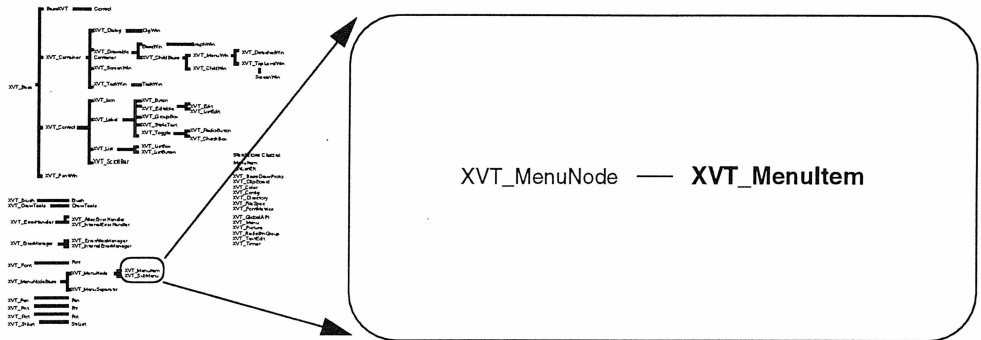
This function is used to construct menus from resources. To create menus at runtime, use Install.

## Implementation Members

```
XVT_Menu( MENU_ITEM* mip )  
GetOwner  
SetOwner  
ConvertTo  
Owner  
List  
InitMenu
```



# XVT\_MenuItem



## Overview

Header File	menu.h
Source File	menu.cc
Superclass	XVT_MenuNode
Subclasses	
Usage	Abstract

The XVT\_MenuItem class specifies the interface to all menu items.

You use this class by creating a subclass that overrides the virtual event handling member function with an implementation that actually does something in response to menu selection.

## Example

See the example in the description of XVT\_Menu.

## Constructors

```

XVT_MenuItem(
    MENU_TAG tag = 0,
    BOOLEAN enabled = TRUE,
    BOOLEAN checked = FALSE,
    BOOLEAN checkable = FALSE,
    const char* text = NULL,
    short mkey = 0 )
XVT_MenuItem( const XVT_MenuItem& item )
~XVT_MenuItem()

```

## Member Functions

---

### XVT\_MenuItem::e\_action

RECEIVE NOTIFICATION OF MENU SELECTION

---

#### Prototypes

```

virtual
void e_action(
    BOOLEAN          shift,
    BOOLEAN          control )

```

#### Parameters

**shift**  
A flag that is TRUE if the shift key was depressed when this item was selected, FALSE if not.

**control**  
A flag that is TRUE if the control key was depressed when this item was selected, FALSE if not.

#### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to the selection of this menu item.

---

## XVT\_MenuItem::GetCheckableState

DETERMINE IF A MENU ITEM IS CHECKABLE

---

### Prototypes

```
BOOLEAN  
GetCheckableState() const
```

### Return Value

A flag that is TRUE if the menu item is checkable, FALSE if not.

---

## XVT\_MenuItem::GetCheckedState

DETERMINE IF A MENU ITEM IS CHECKED

---

### Prototypes

```
BOOLEAN  
GetCheckedState() const
```

### Return Value

A flag that is TRUE if the menu item is checked, FALSE if not.

---

## XVT\_MenuItem::SetCheckedState

CHECK OR UNCHECK A MENU ITEM

---

### Prototypes

```
void  
SetCheckedState(  
    BOOLEAN  
                state )
```

### Parameters

state  
A flag that is TRUE if the menu item is to be checked, FALSE if it is to be unchecked.

### Equivalent C Function

```
win_menu_check()
```

## Implementation Members

XVT\_MenuItem( MENU\_ITEM\* mip )  
ConvertTo  
CopyState  
CheckProtocol  
CheckedState  
CheckableState  
InitProtocols  
KillProtocols

## Inherited Member Functions

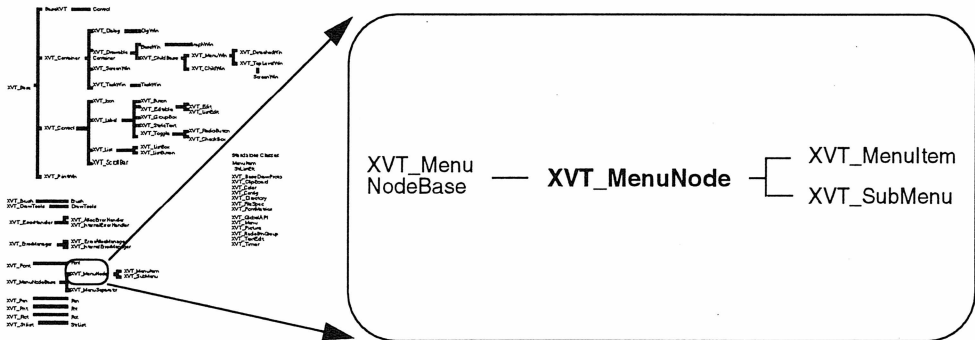
### From XVT\_MenuNode

*page 278*    `BOOLEAN GetEnabledState()`  
*page 278*    `short GetMKey()`  
*page 278*    `void GetTitle( char *buffer, long len )`  
*page 279*    `void SetEnabledState( BOOLEAN state )`  
*page 279*    `void SetTitle( char *str )`

### From XVT\_MenuNodeBase

*page 281*    `virtual XVT_MenuItem *CastToMenuItem()`  
*page 281*    `virtual XVT_MenuNode *CastToMenuNode()`  
*page 281*    `virtual XVT_MenuSeparator *CastToMenuSeparator()`  
*page 281*    `virtual XVT_SubMenu *CastToSubMenu()`  
*page 282*    `XVT_Menu *GetParent()`

# XVT\_MenuNode



## Overview

<b>Header File</b>	menu.h
<b>Source File</b>	menu.cc
<b>Superclass</b>	XVT_MenuNodeBase
<b>Subclasses</b>	XVT_MenuItem, XVT_SubMenu
<b>Usage</b>	Implementation

This class defines the interface common to all menu items that have titles.

## Member Functions

---

### XVT\_MenuNode::GetEnabledState

DETERMINE IF A MENU IS ENABLED OR DISABLED

---

#### Prototypes

```
BOOLEAN  
GetEnabledState() const
```

#### Return Value

A flag that is TRUE if the menu is enabled, FALSE if it is disabled.

---

### XVT\_MenuNode::GetMKey

RETRIEVE THE MENU'S ACCELERATOR KEY

---

#### Prototypes

```
short  
GetMKey() const
```

#### Return Value

The menu's accelerator key code.

---

### XVT\_MenuNode::GetTitle

RETRIEVE A MENU ITEM'S TITLE

---

#### Prototypes

```
BOOLEAN  
GetTitle(  
    char*          unsigned long    buffer,  
    len ) const
```

**Parameters**

`buffer`  
Storage to receive the item's title.

`len`  
A pointer to the length of buffer.

**Return Value**

TRUE if the length of `buffer` was sufficient to hold the item's title,  
FALSE if not. If FALSE is returned, `len` is set to the required length.

---

**XVT\_MenuNode::SetEnabledState**

ENABLE OR DISABLE A MENU ITEM

---

**Prototypes**

```
void
SetEnabledState(
    BOOLEAN          state )
```

**Parameters**

`state`  
A flag that is TRUE if the menu is to be enabled, FALSE if it is to be disabled.

**Description**

Enables or disables a menu item.

**Equivalent C Function**

`win_menu_enable()`

---

**XVT\_MenuNode::SetTitle**

SET A MENU ITEM'S TITLE

---

**Prototypes**

```
void
SetTitle(
    char*          str )
```

**Parameters**

str  
The new title.

**Description**

Sets a menu item's title.

**Equivalent C Function**

win\_set\_menu\_text()

**Implementation Members**

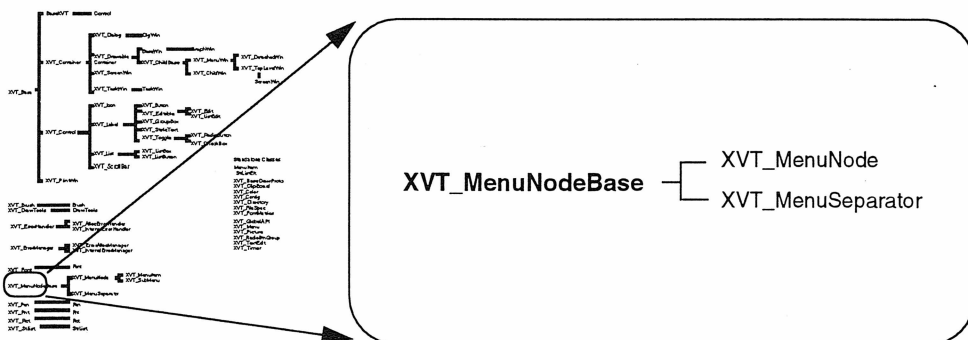
XVT\_MenuNode( MENU\_ITEM\* mip )  
 ConvertTo  
 SetOwner  
 GetTag  
 EnableProtocol  
 TitleProtocol  
 MKey  
 Title  
 Tag  
 EnabledState  
 KillProtocols  
 InitProtocols

**Inherited Member Functions****From XVT\_MenuNodeBase**

*page 281*    virtual XVT\_MenuItem \*CastToMenuItem()  
*page 281*    virtual XVT\_MenuNode \*CastToMenuNode()  
*page 281*    virtual XVT\_MenuSeparator \*CastToMenuSeparator()  
*page 281*    virtual XVT\_SubMenu \*CastToSubMenu()  
*page 282*    XVT\_Menu \*GetParent()



## XVT\_MenuNodeBase



## Overview

<b>Header File</b>	menu.h
<b>Source File</b>	menu.cc
<b>Superclass</b>	
<b>Subclasses</b>	XVT_MenuNode, XVT_MenuSeparator
<b>Usage</b>	Implementation

This class defines the interface common to all menu nodes (items).

## Casts

Virtual cast functions are provided to allow type-safe downcasting. The default implementation of each cast function is to return `NULL`. Each subclass overrides the corresponding cast function to return a pointer to this instead.

```
virtual XVT_MenuItem* CastToMenuItem()
virtual XVT_MenuSeparator* CastToMenuSeparator()
virtual XVT_SubMenu* CastToSubMenu()
virtual XVT_MenuNode* CastToMenuNode()
virtual XVT_DefaultMenuItem* CastToDefaultMenuItem()
```

## Member Functions

---

### XVT\_MenuNodeBase::GetParent

RETRIEVE THE PARENT MENU OF THIS ITEM

---

#### Prototypes

```
XVT_Menu*  
GetParent() const
```

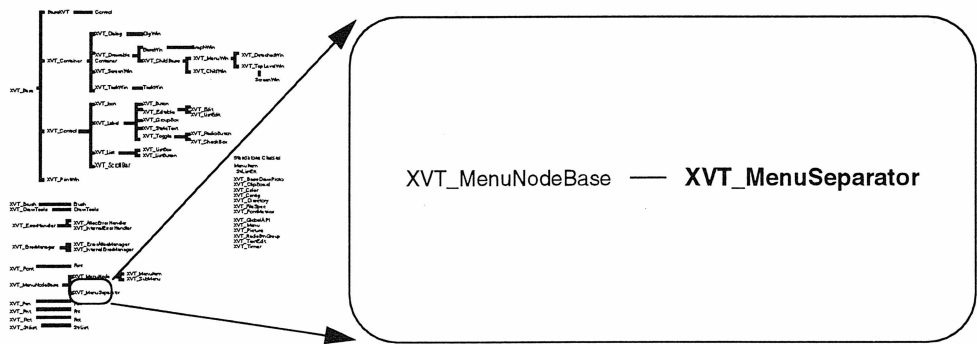
#### Return Value

The parent menu of this item, or NULL if this item is not part of a sub-menu.

## Implementation Members

```
ConvertTo  
SetOwner  
GetOwner  
Owner  
SetParent  
Parent  
NotPortableInfo  
DeleteNPInfo  
WriteNPInfo  
WriteNPInfo  
ReadNPInfo  
CopyNPInfo
```

# XVT\_MenuSeparator



## Overview

Header File	menu.h
Source File	menu.cc
Superclass	XVT_MenuNodeBase
Subclasses	
Usage	Concrete

Instances of this class represent menu separators.

## Constructors

XVT\_MenuSeparator()  
~XVT\_MenuSeparator()

## Implementation Members

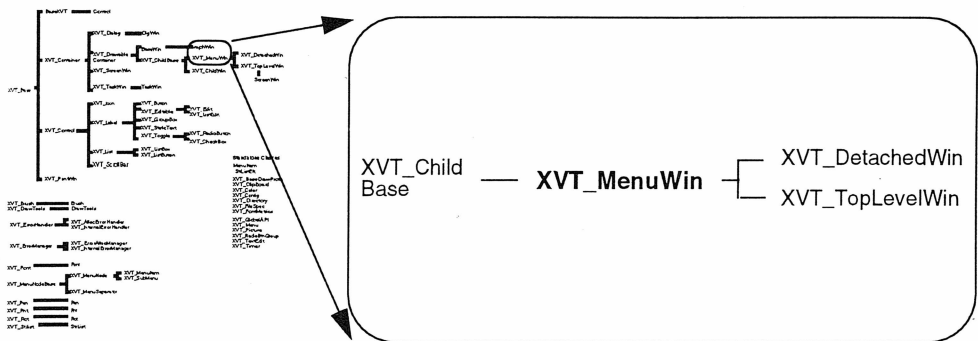
XVT\_MenuSeparator( MENU\_ITEM\* mip )  
ConvertTo

## Inherited Member Functions

### From XVT\_MenuNodeBase

*page 281*    virtual XVT\_MenuItem \*CastToMenuItem()  
*page 281*    virtual XVT\_MenuNode \*CastToMenuNode()  
*page 281*    virtual XVT\_MenuSeparator \*CastToMenuSeparator()  
*page 281*    virtual XVT\_SubMenu \*CastToSubMenu()  
*page 282*    XVT\_Menu \*GetParent()

# XVT\_MenuWin



## Overview

<b>Header File</b>	menuwin.h
<b>Source File</b>	menuwin.cc
<b>Superclass</b>	XVT_ChildBase
<b>Subclasses</b>	XVT_DetachedWin, XVT_TopLevelWin
<b>Usage</b>	Implementation

The menu window class defines the interface common to all windows that have menus.

## Member Variables

---

### XVT\_MenuWin::Menu

A POINTER TO THE WINDOW'S MENU

---

#### Declaration

```
protected:  
  
XVT_Menu* Menu;
```

#### Description

A pointer to the window's menu. Typically, you use this member when replacing default menu items with you own in a window's `e_create` implementation.

## Member Functions

---

### XVT\_MenuWin::e\_close

RECEIVE NOTIFICATION OF A CLOSE REQUEST

---

#### Prototypes

```
virtual void  
e_close()
```

#### Description

This member function must be overridden by a window subclass if the application wishes to take any actions in response to a close request from the user.

A call to `e_close` is generated whenever the user tries to close the window by some means other than selecting close on the file menu (which generates an `e_action` call on the appropriate menu item).

When this event is received, the window hasn't actually been closed; your application must explicitly call `Close` to accomplish that. Additional event handler member functions (such as `e_focus`) can then be called for the window, and your application must be

prepared to handle them. The last event handler member function called for a window is `e_destroy`.

If the `e_close` implementation does not call `Close`, then the window will not be closed, and nothing in the application will change. This distinction is important. Typically, a window will check its state when `e_close` is called. If the state indicates that the contents of the window have been saved (for example), then the application may simply call `Close`. If, however, the contents have not been saved, the application may display a dialog asking if the user wishes to save or discard changes, so that the changes may be preserved before the call to `Close` is made.

---

## XVT\_MenuWin::e\_font

RECEIVE NOTIFICATION OF A FONT CHANGE

---

### Prototypes

```
virtual void
e_font(
    XVT_Font      font,
    FONT_PART     part )
```

### Parameters

`font`  
The new font.

`part`  
The part of the font which changed.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to font changes involving the window.

---

## XVT\_MenuWin::GetMenu

RETRIEVE THE CURRENT MENU

---

### Prototypes

```
XVT_Menu*
GetMenu()
```

**Return Value**

A pointer to the menu currently attached to the window.

**Description**

Dissociates the menu from the window and returns a pointer to the menu. You must either delete it or give it back to SetMenu.

**Equivalent C Function**

win\_menu\_fetch()

---

## XVT\_MenuWin::GetTitle

RETRIEVE THE WINDOW'S TITLE

---

**Prototypes**

```
BOOLEAN  
GetTitle(  
    char*                buffer,  
    unsigned long*       len ) const
```

**Parameters**

buffer  
Storage for the title string.

len  
A pointer to the length of buffer.

**Return Value**

TRUE if the length of buffer was sufficient to hold the application's name, FALSE if not. If FALSE is returned, len is set to the required length.

**Description**

Retrieves the window's current title. The title retrieved reflects whatever is stored in the native window. In particular, if the native window has truncated the title, or if text was added to the title by SetDocTitle, those modifications will be present in the title returned.

**Equivalent C Function**

get\_title()



---

## XVT\_MenuWin::SetDocTitle

SET A WINDOW'S TITLE

---

### Prototypes

```
void  
SetDocTitle(  
    const char*          str )
```

### Parameters

str  
A null-terminated string containing the window's new title.

### Description

This function is similar to `SetTitle`, differing only in that it takes the given string and forms a title that conforms to the native style guidelines for document windows. Typically the application name given in the `XVT_Config` instance is prepended to the title.

### Equivalent C Function

```
set_doc_title()
```

---

## XVT\_MenuWin::SetFontMenu

SET FONT MENU CHECKMARKS

---

### Prototypes

```
void  
SetFontMenu(  
    XVT_Font          font )
```

### Parameters

font  
The selected font that is to be reflected in the font menu. A null pointer indicates that no font is to be selected.

### Description

This function makes the font menu show the font and style given by font as being selected.

If your application is one where a single font is used throughout the entire window (e.g., a text editor), then you should set the font menu to the `XVT_Font` displayed in the window. If, however, your

application allows for the display and selection of different text objects drawn with different fonts, then it should set the font menu check marks to match the XVT\_Font used in drawing the currently selected item. If there is no currently selected item, then the font menu should either be completely unchecked, or should be set to the XVT\_Font that would be used if a new text item were created.

### Implementation Notes

On the Mac, the point sizes on the Style menu that correspond to the real fonts that are available are outlined, instead of checked.

### Equivalent C Function

`win_set_font_menu()`

---

## XVT\_MenuWin::SetMenu

SET THE CURRENT MENU

---

### Prototypes

```
XVT_Menu*
SetMenu(
    XVT_Menu*      menu )
```

### Parameters

`menu`

The menu that will become the window's menu when the SetMenu call completes.

### Return Value

The menu replaced by `menu`. You must delete the old menu.

### Description

Replaces a window's menu with the menu specified by `menu`. The menu pointed to by `menu` is copied, not consumed.

### Equivalent C Function

`win_menu_show()`

---

## XVT\_MenuWin::SetTitle

SET A WINDOW'S TITLE

---

### Prototypes

```
void
SetTitle(
    const char*      str )
```

### Parameters

**str**  
A null-terminated string containing the window's new title.

### Description

Modifies the title field of the window to display the title passed in **str**.

### Equivalent C Function

```
set_title()
```

## Implementation Members

```
GetMenuNode
TitleProtocol
DoInit
CommandEvent
```

## Inherited Member Functions

### From XVT\_ChildBase

<i>page 49</i>	virtual void e_hscroll( SCROLL_CONTROL activity, short pos )
<i>page 49</i>	virtual void e_vscroll( SCROLL_CONTROL activity, short pos )
<i>page 50</i>	XVT_TextEdit* GetActiveTextEdit()
<i>page 50</i>	XVT_Pnt GetCaretPos() const
<i>page 51</i>	BOOLEAN GetCaretState() const
<i>page 51</i>	BOOLEAN GetEnabledState()
<i>page 51</i>	XVT_ChildBase *GetParent() const

<i>page 52</i>	<code>long GetScrollPosition( SCROLL_TYPE scroll_type ) const</code>
<i>page 52</i>	<code>long GetScrollProportion( SCROLL_TYPE scroll_type ) const</code>
<i>page 53</i>	<code>void GetScrollRange( SCROLL_TYPE scroll_type, long *min, long *max ) const</code>
<i>page 54</i>	<code>XVT_TextEdit* GetTextEdit( long id )</code>
<i>page 54</i>	<code>BOOLEAN GetVisibleState()</code>
<i>page 55</i>	<code>void MakeFront()</code>
<i>page 55</i>	<code>void ReleaseMouse()</code>
<i>page 56</i>	<code>void SetCaretDimensions( XVT_Pnt vector )</code>
<i>page 56</i>	<code>void SetCaretPos( XVT_Pnt point )</code>
<i>page 57</i>	<code>void SetCaretState( BOOLEAN state )</code>
<i>page 57</i>	<code>void SetCursor( CURSOR cursor )</code>
<i>page 58</i>	<code>void SetEnabledState( BOOLEAN state )</code>
<i>page 59</i>	<code>void SetScrollPosition( SCROLL_TYPE scroll_type, long position )</code>
<i>page 60</i>	<code>void SetScrollProportion( SCROLL_TYPE scroll_type, long proportion )</code>
<i>page 60</i>	<code>void SetScrollRange( SCROLL_TYPE scroll_type, long min, long max, long pos )</code>
<i>page 61</i>	<code>void SetVisibleState( BOOLEAN f )</code>
<i>page 62</i>	<code>void TrapMouse()</code>

### **From XVT\_DrawableContainer**

<i>page 129</i>	<code>void Clear()</code>
<i>page 129</i>	<code>void Clear( XVT_Color color )</code>
<i>page 129</i>	<code>void Close()</code>
<i>page 128</i>	<code>XVT_BaseDrawProto* DrawProtocol</code>
<i>page 130</i>	<code>virtual void e_char( short chr, BOOLEAN shift, BOOLEAN control)</code>
<i>page 131</i>	<code>virtual void e_create()</code>
<i>page 132</i>	<code>virtual void e_destroy()</code>

<i>page 132</i>	virtual void e_focus( BOOLEAN active )
<i>page 133</i>	virtual void e_mouse_dbl( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 134</i>	virtual void e_mouse_down( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_move( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_up( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 136</i>	virtual void e_size( XVT_Rct boundary )
<i>page 137</i>	virtual void e_timer( long id )
<i>page 137</i>	virtual void e_update( XVT_Rct boundary )
<i>page 139</i>	virtual long e_user( long id, void *data )
<i>page 140</i>	XVT_Control *GetCtl( long cid )
<i>page 140</i>	long GetCtlCount()
<i>page 141</i>	EVENT_MASK GetEventMask() const
<i>page 141</i>	XVT_Control *GetFirstCtl()
<i>page 142</i>	XVT_ChildBase *GetFirstWin()
<i>page 142</i>	XVT_Control *GetNextCtl()
<i>page 143</i>	XVT_ChildBase *GetNextWin()
<i>page 143</i>	long GetWinCount()
<i>page 144</i>	void Invalidate()
<i>page 144</i>	void Invalidate( XVT_Rctregion )
<i>page 145</i>	void Scroll( XVT_Rct boundary, long dh, long dv )

*page 146*    void SetEventMask( EVENT\_MASK ask )

*page 148*    void SetInnerRect( XVT\_Rct r )

### **From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()

*page 10*    virtual DlgWin\* CastToDlgWin()

*page 10*    virtual ScreenWin\* CastToScreenWin11()

*page 10*    virtual TaskWin\* CastToTaskWin11()

*page 11*    virtual XVT\_Button \*CastToButton()

*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()

*page 11*    virtual XVT\_ChildWin \*CastToChildWin()

*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()

*page 11*    virtual XVT\_Dialog \*CastToDialog()

*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()

*page 11*    virtual XVT\_Edit \*CastToEdit()

*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()

*page 11*    virtual XVT\_Icon \*CastToIcon()

*page 11*    virtual XVT\_ListBox \*CastToListBox()

*page 11*    virtual XVT\_ListButton \*CastToListButton()

*page 11*    virtual XVT\_ListEdit \*CastToListEdit()

*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()

*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()

*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()

*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()

*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()

*page 11*    virtual XVT\_StaticText \*CastToStaticText()

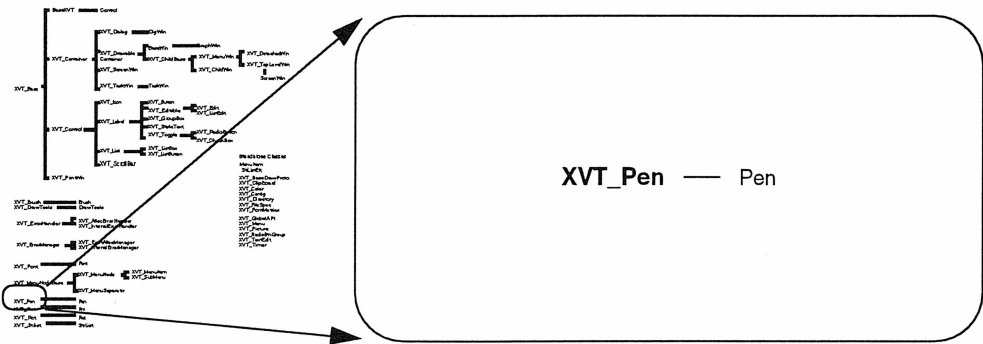
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()

*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()

*page 12*    virtual XVT\_Rct GetInnerRect()

*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_Pen



## Overview

Header File	tools.h
Source File	tools.cc
Superclass	
Subclasses	Pen
Usage	Concrete

Instances of the pen class define how outlines of drawing primitives may be rendered.

## Constructors

```
XVT_Pen()
XVT_Pen( short width, PAT_STYLE pattern,
         PEN_STYLE style, XVT_Color color )
    Create a new pen with the given width, pattern, style, and color.
    Equivalent to using the default constructor followed by
    SetWidth, SetPattern, SetStyle, and SetColor.
XVT_Pen( const XVT_Pen& pen )
~XVT_Pen()
```

## Operators

```
XVT_Pen& operator=( const XVT_Pen& pen )  
BOOLEAN operator==( const XVT_Pen& pen )  
Pens may be assigned and compared for equality.
```

## Member Functions

---

### XVT\_Pen::GetColor

RETRIEVE A PEN'S COLOR

---

#### Prototypes

```
XVT_Color  
GetColor() const
```

#### Return Value

The pen's current color.

---

### XVT\_Pen::GetPattern

RETRIEVE A PEN'S PATTERN

---

#### Prototypes

```
PAT_STYLE  
GetPattern() const
```

#### Return Value

The pen's current pattern.

---

### XVT\_Pen::GetStyle

RETRIEVE A PEN'S STYLE

---

#### Prototypes

```
PEN_STYLE  
GetStyle() const
```



**Return Value**

The pen's current style.

---

**XVT\_Pen::GetWidth**

RETRIEVE A PEN'S WIDTH

---

**Prototypes**

```
short  
GetWidth() const
```

**Return Value**

The pen's width.

---

**XVT\_Pen::SetColor**

SET A PEN'S COLOR

---

**Prototypes**

```
void  
SetColor(  
    XVT_Color          color )
```

**Parameters**

```
color  
    The pen's new color.
```

**Description**

Sets a pen's color.

---

**XVT\_Pen::SetPattern**

SET A PEN'S PATTERN

---

**Prototypes**

```
void  
SetPattern(  
    PAT_STYLE          pattern )
```

**Parameters**

pattern

The new pen pattern.

The following members of the PAT\_STYLE enumeration are valid for pens:

PAT\_SOLID

Produces a solid line.

PAT\_HOLLOW

Produces no outline at all.

PAT\_RUBBER

Produces a grayish or dotted line that conforms to the native window system's look for rubber banding.

**Description**

Sets a pen's pattern.

---

**XVT\_Pen::SetStyle**

SET A PEN'S STYLE

---

**Prototypes**

```
void  
SetStyle(  
    PEN_STYLE          style )
```

**Parameters**

style

The new pen style.

Members of the PEN\_STYLE enumeration are:

P\_SOLID

Produces a solid line.

P\_DOT

Produces a dotted line.

P\_DASH

Produces a dashed line.

**Description**

Sets a pen's style.

---

## XVT\_Pen::SetWidth

SET A PEN'S WIDTH

---

### Prototypes

```
void  
SetWidth(  
    short                width )
```

### Parameters

width  
The new pen width.

### Description

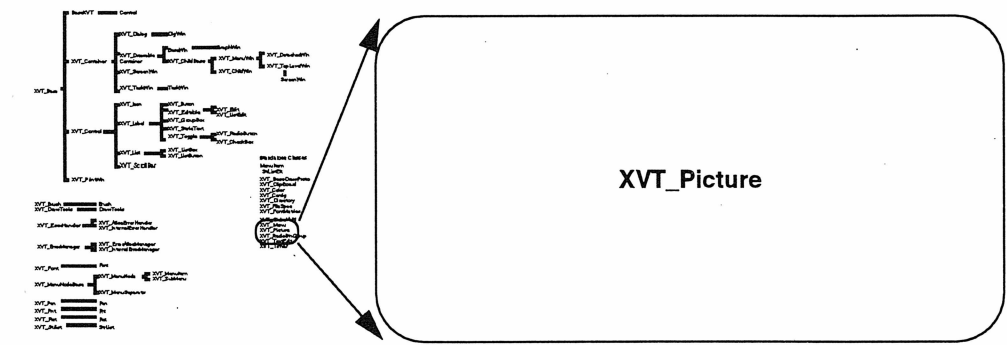
Sets a pen's width.

A pen's width is the width in pixels of the line produced by a pen stroke.

## Implementation Members

ConvertTo  
ConvertFrom  
Width  
Pattern  
Style  
Color

# XVT\_Picture



## Overview

Header File	picture.h
Source File	picture.cc
Superclass	
Subclasses	
Usage	Concrete

Instances of XVT\_Picture represent collections of drawing primitives as drawn to a particular window. Pictures may be recorded in any drawable window, written to the clipboard, or converted into opaque data.

## Example

The following code shows how to create a picture in a window:

```
{
    XVT_Picture* myPicture;
    XVT_DrawableContainer* myWindow;
    XVT_RCT itsClientArea;
    .
    .
    .

    itsClientArea = myWindow->GetInnerRect();

    new XVT_Picture(
        myWindow,
        itsClientArea.Normalize() )

    // drawing operations in myWindow are recorded
    // in myPicture
    .
    .
    .

    myPicture->Close()

    // myPicture is now ready to use, it may be drawn
    // in any drawable or converted into opaque data.
}
```

## Constructors

```
XVT_Picture( XVT_DrawableContainer *drawable,
             XVT_Rct boundary )
    Create a new picture in a window. The new picture captures all
    drawing done in drawable that is inside boundary until the
    Close member function is called. After the c member function
    as been called, the picture can be drawn or turned into opaque
    data using GetOpaqueData.
    Equivalent to picture_open.

XVT_Picture( const char* buffer, long size,
             XVT_Rct boundary )
    Create a picture from opaque data.
    Equivalent to picture_make.

~XVT_Picture()
```

## Member Functions

---

### XVT\_Picture::Close

STOP RECORDING DRAWING PRIMITIVES

---

#### Prototypes

```
void  
Close()
```

#### Description

Stops recording drawing primitives and creates the picture.

This member function causes an error unless the picture was created with the `XVT_Picture( XVT_DrawableContainer *drawable, XVT_Rct boundary )` constructor.

#### Equivalent C Function

```
picture_close()
```

---

### XVT\_Picture::GetLockedState

DETERMINE IF THE PICTURE IS CURRENTLY LOCKED

---

#### Prototypes

```
BOOLEAN  
GetLockedState() const
```

#### Return Value

TRUE if the picture is currently locked, FALSE if not.

---

### XVT\_Picture::GetOpaqueData

CONVERT A PICTURE INTO OPAQUE DATA

---

#### Prototypes

```
void  
GetOpaqueData(  
    char*  
    buffer ) const
```

**Parameters**

buffer

A buffer to hold the picture data. The buffer must be at least as big as the number returned by GetOpaqueDataSize.

**Description**

Converts a picture into opaque data. The opaque picture data is not portable. If you write picture data to a file you will *not* be able to read that data in and create a picture on any platform with a different architecture or window system.

**Implementation Notes**

XVT/Mac

A picture is a PICT.

XVT/Win, XVT/PM

A picture is a bitmap.

XVT/CH

A picture is a character map.

**Equivalent C Function**

picture\_lock()

picture\_unlock()

---

## XVT\_Picture::GetOpaqueDataSize

DETERMINE THE SIZE OF BUFFER TO HOLD OPAQUE DATA

---

**Prototypes**

long

GetOpaqueDataSize() const

**Return Value**

The minimum size in bytes of buffer necessary to store an opaque representation of this picture.

**Equivalent C Function**

picture\_lock()

---

## XVT\_Picture::GetOpenState

DETERMINE IF THE PICTURE IS CURRENTLY OPEN

---

### Prototypes

```
BOOLEAN  
GetOpenState() const
```

### Return Value

TRUE if the picture is currently open, FALSE if not.

---

## XVT\_Picture::Lock

LOCK A PICTURE'S DATA

---

### Prototypes

```
BOOLEAN  
Lock()
```

### Return Value

TRUE if the picture was successfully locked, FALSE if not.

### Description

Prepare to obtain the picture's opaque data.

### Equivalent C Function

```
picture_lock()
```

---

## XVT\_Picture::Unlock

UNLOCK A PICTURE'S DATA

---

### Prototypes

```
void  
Unlock()
```

### Return Value

TRUE if the picture was successfully unlocked, FALSE if not.



**Description**

Release a picture's opaque data.

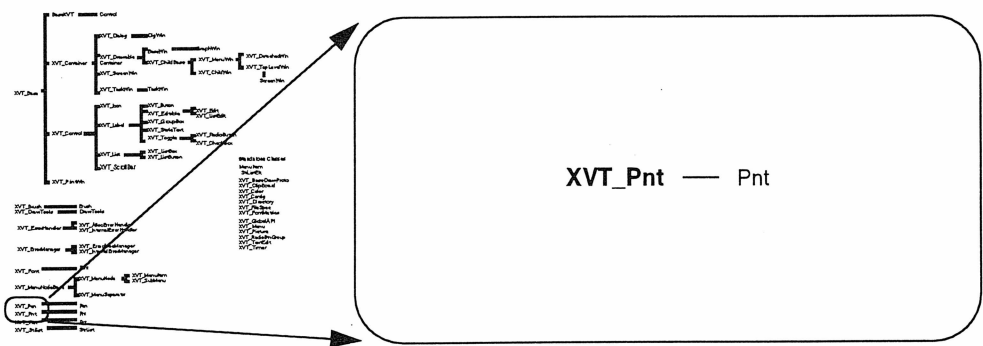
**Equivalent C Function**

picture\_unlock()

**Implementation Members**

GetDrawable  
GetPictData  
PictData  
OpaqueData  
OpaqueDataSize  
Drawable  
OpenState  
LockedState  
DoInit

# XVT\_Pnt



## Overview

Header File	pnt.h
Source File	pnt.cc
Superclass	
Subclasses	Pnt
Usage	Concrete

Instances of the point class model mathematical points.

## Constructors

```
XVT_Pnt( short x = 0, short y = 0 )
XVT_Pnt( const XVT_Pnt& point )
virtual ~XVT_Pnt()
```

## Operators

```
XVT_Pnt& operator=( const XVT_Pnt& point )
BOOLEAN operator==( const XVT_Pnt& point ) const
BOOLEAN operator!=( const XVT_Pnt& point ) const
```

```
BOOLEAN operator>( const XVT_Pnt& point ) const
BOOLEAN operator>=( const XVT_Pnt& point ) const
BOOLEAN operator<( const XVT_Pnt& point ) const
BOOLEAN operator<=( const XVT_Pnt& point ) const
XVT_Pnt operator+( const short offset ) const
XVT_Pnt& operator+=( const short offset )
XVT_Pnt operator+( const XVT_Pnt& point ) const
XVT_Pnt& operator+=( const XVT_Pnt& point )
XVT_Pnt operator-( const short offset ) const
XVT_Pnt& operator-=( const short offset )
XVT_Pnt operator-( const XVT_Pnt& point ) const
XVT_Pnt& operator-=( const XVT_Pnt& point )
XVT_Pnt operator*( const short factor ) const
XVT_Pnt& operator*=( const short factor )
```

## Member Functions

---

### XVT\_Pnt::GetX

RETRIEVE A POINT'S X COORDINATE

---

#### Prototypes

```
virtual short
GetX() const
```

#### Return Value

The point's X coordinate.

---

### XVT\_Pnt::GetY

RETRIEVE A POINT'S Y COORDINATE

---

#### Prototypes

```
virtual short
GetY() const
```

**Return Value**

The point's Y coordinate.

---

**XVT\_Pnt::SetX**

SET A POINT'S X COORDINATE

---

**Prototypes**

```
virtual void  
SetX(  
    short                pos )
```

**Parameters**

pos  
The point's new X coordinate.

**Description**

Sets a point's X coordinate.

---

**XVT\_Pnt::SetY**

SET A POINT'S Y COORDINATE

---

**Prototypes**

```
virtual void  
SetY(  
    short                pos )
```

**Parameters**

pos  
The point's new Y coordinate.

**Description**

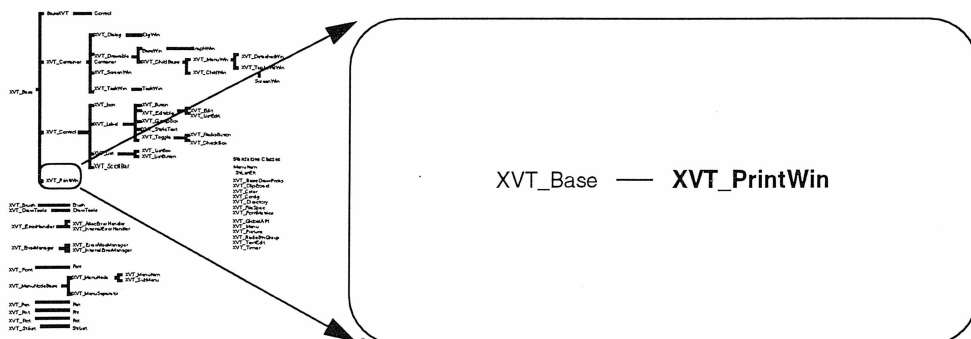
Sets a point's Y coordinate.

---

**Implementation Members**

```
ConvertTo  
ConvertFrom  
X  
Y
```

# XVT\_PrintWin



## Overview

<b>Header File</b>	printwin.h
<b>Source File</b>	printwin.cc
<b>Superclass</b>	XVT_Base
<b>Subclasses</b>	
<b>Usage</b>	Abstract

Print windows are used to structure the interface to printing. Print windows have the same drawing interface as any other drawable window; however, they do not have any event handler member functions. Instead three virtual member functions control the print loop.

Before creating a print window, you should already have invoked PageSetup to set up the printer.

Because the print job can run in a separate thread, there is no way for XVT++ application to know when it has completed. For this reason, the print window must delete itself when the print job has completed. You should never delete a print window, nor should you allocate one by any other means than new.

## Example

```
class MyPrintWin : public XVT_PrintWin
{
    BOOLEAN AnotherPage();
    void DrawInit();
    void DrawAction();
};
.
.
.

{
    XVT_PrintWin* thePrintWin;

    thePrintWin = new MyPrintWin( "My Print Job" );
    thePrintWin->Init();

    // NOTE: do not delete thePrintWin
}
```

## Constructors

```
XVT_PrintWin( const char* jobName = NULL )
virtual ~XVT_PrintWin()
```

## Member Fields

---

### XVT\_PrintWin::DrawProtocol

THE PRINT WINDOW'S DRAWING PROTOCOL

---

#### Prototype

```
XVT_BaseDrawProto*
DrawProtocol
```

#### Description

The drawing protocol provides access to all of the XVT++ drawing functionality. Access to drawing functionality is indirected in this manner so that the drawing code can be made to work for both windows and print windows.

# Member Functions

---

## XVT\_PrintWin::AnotherPage

DETERMINE IF ANOTHER PAGE IS TO BE PRINTED

---

### Prototypes

```
protected:  
  
virtual BOOLEAN  
AnotherPage() = 0
```

### Return Value

A flag that is TRUE if another page is to be printed, FALSE if not. If false is returned, the print job is terminated.

### Description

You must override this function.

The print window calls it immediately before starting each page.

---

## XVT\_PrintWin::DrawAction

DRAW A PRINT BAND

---

### Prototypes

```
protected:  
  
virtual void  
DrawAction() = 0
```

### Description

You must override this function.

The print window calls DrawAction once for each band printed. There is at least one print band per page. Print bands are not necessarily disjoint; some environments will in fact use several bands with identical boundaries, printing different primitives in each band.

---

## XVT\_PrintWin::DrawInit

PREPARE TO PRINT

---

### Prototypes

```
protected:
    virtual void
    DrawInit()
```

### Description

You can override this function.

Your implementation should initialize any context (page number, draw tools, and so on) that you need for the print job.

---

## XVT\_PrintWin::GetOutputFile

RETRIEVE THE OUTPUT FILE NAME

---

### Prototypes

```
#if XVTWS == MTFWS || XVTWS == XOLWS || XVTWS == WMWS
    BOOLEAN
    GetOutputFile(
        char*          buffer,
        unsigned long* len ) const
#endif
```

### Parameters

**buffer**  
Storage to receive the output file name.

**len**  
A pointer to the length of buffer.

### Return Value

TRUE if the length of buffer was sufficient to hold the output file name, FALSE if not. If FALSE is returned, len is set to the required length.



**Implementation Notes**

XVT/XM, XVT/XOL, XVT/CH

This function is available only with XVT/XM, XVT/XOL and XVT/CH. Attempts to use it in other environments will result in a compile-time error.

**Equivalent C Function**

`get_value( ATTR_PS_PRINT_FILE_NAME )`

---

## XVT\_PrintWin::GetPrintRcd

RETRIEVE A COPY OF THE PRINT WINDOW'S PRINT RECORD

---

**Prototypes**

```
void*
GetPrintRcd() const
```

**Return Value**

A pointer to the print window's print record. The record may be copied as opaque data. The size of the record may be obtained by calling `GetPrintRcdSize`.

---

## XVT\_PrintWin::GetPrintRcdSize

RETRIEVE THE SIZE OF THE PRINT RECORD

---

**Prototypes**

```
long
GetPrintRcdSize() const
```

**Return Value**

This size of the print record in bytes.

---

## XVT\_PrintWin::Init

INITIALIZE A PRINT WINDOW

---

### Prototypes

```
BOOLEAN
Init()
```

### Description

Creates the underlying print window and starts printing. If supported, printing takes place in a separate thread.

### Implementation Notes

XVT/PM

Since printing uses a separate thread, you need to take care that ongoing modifications to the model being printed do not corrupt the print thread.

### Equivalent C Function

```
start_print_thread()
```

---

## XVT\_PrintWin::SetOutputFile

SET THE OUTPUT FILE NAME

---

### Prototypes

```
#if XVTWS == MTFWS || XVTWS == XOLWS || XVTWS == WMWS
void
SetOutputFile(
    const char*      name )
#endif
```

### Parameters

name  
The new output file name.

### Description

Sets the output file name.

## Implementation Notes

XVT/XM, XVT/XOL, XVT/CH

This function is available only with XVT/XM, XVT/XOL and XVT/CH. Attempts to use it in other environments will result in a compile-time error.

## Equivalent C Function

```
set_value( ATTR_PS_PRINT_FILE_NAME )
```

---

# XVT\_PrintWin::SetPrintRcd

SET THE PRINT WINDOW'S PRINT RECORD

---

## Prototypes

```
void  
SetPrintRcd(  
    void* print_rcd )
```

## Parameters

`print_rcd`  
A pointer to the opaque print record structure. It will be copied, not consumed.

## Description

Set the print window's print record. The print record provides additional parameters, such as page orientation, to the native print driver.

---

# XVT\_PrintWin::ValidatePrintRcd

VALIDATE THE PRINT RECORD

---

## Prototypes

```
BOOLEAN  
ValidatePrintRcd()
```

## Return Value

TRUE if the print record was modified and should be saved, FALSE if not.

## Description

This function ensures that the current print record is valid for the current system configuration. You use this function after setting a print record read from a document file.

## Implementation Notes

### XVT/Mac

The current printer is chosen with the Chooser Desk Accessory. This function will validate a print record against the current printer.

### XVT/Win, XVT/PM

The current printer is stored as a part of the print record. This function will make sure that that printer exists and that its settings are valid.

## Implementation Members

CloseProtocol  
PrintRct  
Size  
Title  
InitProtocols

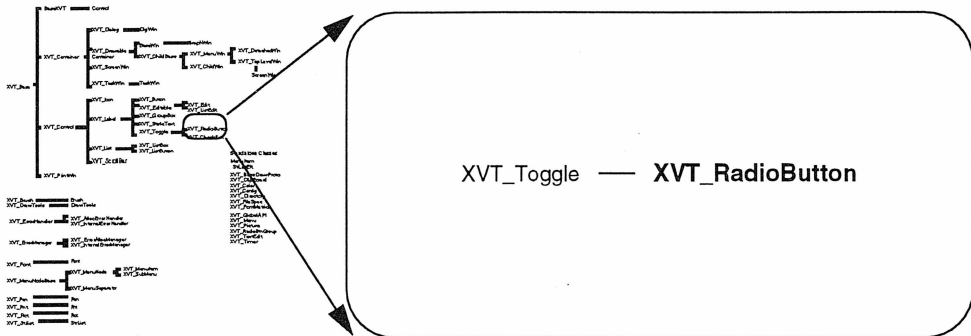
## Inherited Member Functions

### From XVT\_Base

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()

<i>page 11</i>	virtual XVT_Edit *CastToEdit()
<i>page 11</i>	virtual XVT_GroupBox *CastToGroupBox()
<i>page 11</i>	virtual XVT_Icon *CastToIcon()
<i>page 11</i>	virtual XVT_ListBox *CastToListBox()
<i>page 11</i>	virtual XVT_ListButton *CastToListButton()
<i>page 11</i>	virtual XVT_ListEdit *CastToListEdit()
<i>page 11</i>	virtual XVT_MenuWin *CastToMenuWin()
<i>page 11</i>	virtual XVT_PrintWin *CastToPrintWin()
<i>page 11</i>	virtual XVT_RadioButton *CastToRadioButton()
<i>page 11</i>	virtual XVT_ScreenWin *CastToScreenWin()
<i>page 11</i>	virtual XVT_ScrollBar *CastToScrollBar()
<i>page 11</i>	virtual XVT_StaticText *CastToStaticText()
<i>page 11</i>	virtual XVT_TaskWin *CastToTaskWin()
<i>page 11</i>	virtual XVT_TopLevelWin *CastToTopLevelWin()
<i>page 12</i>	virtual XVT_Rct GetInnerRect()
<i>page 13</i>	virtual XVT_Rct GetOuterRect()

# XVT\_RadioButton



## Overview

<b>Header File</b>	radiobtn.h
<b>Source File</b>	radiobtn.cc
<b>Superclass</b>	XVT_Toggle
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The `XVT_RadioButton` class specifies the interface to radio buttons.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

Radio buttons are similar to check boxes. The user can turn radio buttons on or off like check boxes. However, radio buttons are different from check boxes in two respects: only one radio button in the group can be on (the rest must be off), and radio buttons have a different shape.

Radio buttons are always used in conjunction with a radio button group, which makes sure that only one button at a time is depressed. Radio button groups are instances of `XVT_RadioButtonGroup`. Other than the constructor, radio button groups have no public interface.

## Example

Here is how to create a group of three radio buttons:

```
MyWin::e_create()
{
    XVT_RadioBtnGroup* theGroup;
    XVT_RadioButton* theNewButton;
    XVT_Rct buttonBoundary =
        XVT_Rct( 100, 100 200, 124 );
    XVT_Pnt buttonOffset = XVT_Pnt( 0, 24 );

    theGroup = new XVT_RadioButtonGroup;

    theNewButton = new MyRadio( this, theGroup, 1001 );
    theNewButton->Init(
        buttonBoundary,
        0,
        "choice 1" );

    buttonBoundary += buttonOffset;
    theNewButton = new MyRadio( this, theGroup, 1002 );
    theNewButton->Init(
        buttonBoundary,
        0,
        "choice 2" );

    buttonBoundary += buttonOffset;
    theNewButton = new MyRadio( this, theGroup, 1003 );
    theNewButton->Init(
        buttonBoundary,
        0,
        "choice 3" );
}
```

Note that the control IDs must still be sequential integers.

## Constructors

```
XVT_RadioButton(
    XVT_Dialog* parent,
    XVT_RadioBtnGroup* group,
    long id )
XVT_RadioButton(
    XVT_DrawableContainer* parent,
    XVT_RadioBtnGroup* group,
    long id )
    Create a radio button in the given radio button group.
XVT_RadioBtnGroup()
    Create a radio button group. The only use for a radio button
    group is as an argument to a radio button constructor.
virtual ~XVT_RadioButton()
```

## Member Functions

---

### XVT\_RadioButton::SetCheckedState

---

CHECK OR UNCHECK A RADIO BUTTON

---

#### Prototypes

```
void
SetCheckedState()
```

#### Description

Check a radio button and uncheck all other radio buttons in its group.

#### Equivalent C Function

```
win_check_radio_button()
```

## Implementation Members

Group

## Inherited Member Functions

#### From XVT\_Toggle

*page 394*    virtual void e\_action()

*page 394*    virtual BOOLEAN GetCheckedState()

#### From XVT\_Label

*page 239*    void GetTitle( char\* str, unsigned long\* len )

*page 239*    virtual BOOLEAN Init( XVT\_Rct boundary, long = 0L, char \*  
= NULL )

*page 240*    void SetTitle( char\* str )

#### From XVT\_Control

*page 92*    virtual void Close()

*page 93*    virtual void e\_create()

*page 93*    virtual void e\_destroy()



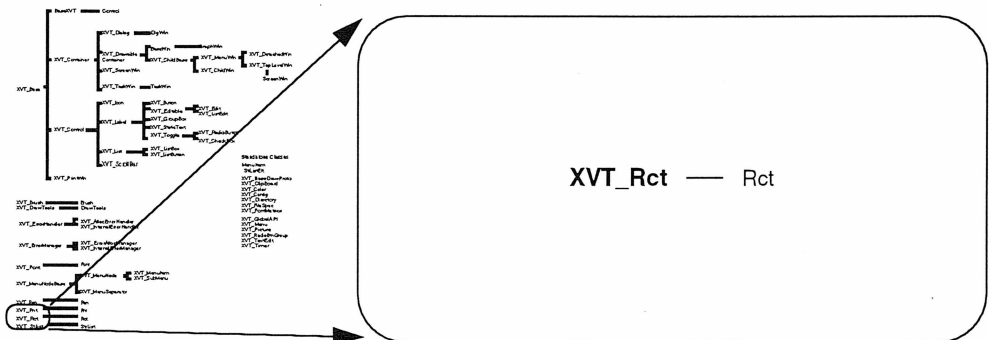
*page 94*    virtual long e\_user( long id, void \*data )  
*page 95*    BOOLEAN GetEnabledState()  
*page 95*    long GetID( void )  
*page 95*    XVT\_Base \*GetParent( void )  
*page 96*    BOOLEAN GetVisibleState()  
*page 96*    void Init()  
*page 96*    void MakeFront()  
*page 97*    void SetEnabledState( BOOLEAN state )  
*page 98*    void SetInnerRect( XVT\_Rct boundary )  
*page 98*    void SetVisibleState( BOOLEAN state )

### **From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()

<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>

# XVT\_Rct



## Overview

<b>Header File</b>	rct.h
<b>Source File</b>	rct.cc
<b>Superclass</b>	
<b>Subclasses</b>	Rct
<b>Usage</b>	Concrete

Instances of this class model mathematical rectangles.

## Constructors

```

XVT_Rct()
XVT_Rct( XVT_Pnt top_left, XVT_Pnt bottom_right )
XVT_Rct( long top, long left, long bottom, long right )
XVT_Rct( XVT_Pnt top_left, long width, long height )
XVT_Rct( const XVT_Rct& rect )
~XVT_Rct()

```

## Operators

```

XVT_Rct& operator=( const XVT_Rct& rect )
BOOLEAN operator==( const XVT_Rct& rect ) const
BOOLEAN operator!=( const XVT_Rct& rect ) const
XVT_Rct operator+( const XVT_Pnt& point ) const
XVT_Rct& operator+=( const XVT_Pnt& point )
XVT_Rct operator+( const short offset ) const
XVT_Rct& operator+=( const short offset )

```

## Member Functions

---

### XVT\_Rct::Constrain

---

CONSTRAIN A POINT TO BE INSIDE A RECTANGLE

---

#### Prototypes

```

XVT_Pnt
Constrain(
    XVT_Pnt          point ) const

```

#### Parameters

point  
The point to be constrained to the rectangle.

#### Return Value

The point closest to point which is inside the rectangle.

#### Description

Constrains a point to lie within a rectangle.

## XVT\_Rct::Contains

DETERMINE IF A RECTANGLE CONTAINS A POINT OR ANOTHER RECTANGLE

### Prototypes

```

BOOLEAN
Contains(
    XVT_Rct          rect ) const

BOOLEAN
Contains(
    XVT_Pnt          point ) const

```

### Parameters

rect  
A rectangle.

point  
A point.

### Return Value

A flag that is TRUE if the argument object is contained by the rectangle, FALSE if not.

### Description

A point (x, y) is contained in a rectangle ((x<sub>ul</sub>, y<sub>ul</sub>), (x<sub>lr</sub>, y<sub>lr</sub>)) if the following conditions are met:

$$x_{ul} \leq x < x_{lr}$$

$$y_{ul} \leq y < y_{lr}$$

A rectangle is contained by another rectangle if both corner points are contained.

Contains( rect )  
Determines if the rectangle contains another rectangle.

Contains( point )  
Determines if a rectangle contains a point.

### Equivalent C Function

pt\_in\_rect()

---

## XVT\_Rct::Difference

COMPUTE THE DIFFERENCE OF TWO RECTANGLES

---

### Prototypes

```
short  
Difference(  
    XVT_Rct&  
    XVT_Rct  
    boundary,  
    list[] ) const
```

### Parameters

**boundary**  
The subtrahend rectangle.

**list**  
An array of four rectangles to receive the difference rectangles.

### Return Value

The number of rectangles in the difference. The difference between two rectangles can always be expressed as the union of between 0 and 4 rectangles. If there are 0 rectangles in the difference it indicates that this and boundary are the same.

### Description

Computes the difference of two rectangles. The difference is defined as the area that is in this but not in boundary.

---

## XVT\_Rct::GetBottomLeft

RETRIEVE THE LOWER-LEFT CORNER OF A RECTANGLE

---

### Prototypes

```
XVT_Pnt  
GetBottomLeft() const
```

### Return Value

The point at the lower-left corner of the rectangle.

---

## XVT\_Rct::GetBottomRight

RETRIEVE THE LOWER-RIGHT CORNER OF A RECTANGLE

---

### Prototypes

```
XVT_Pnt  
GetBottomRight() const
```

### Return Value

The point at the lower-right corner of the rectangle.

---

## XVT\_Rct::GetDimVect

RETRIEVE A RECTANGLE'S DIMENSIONS

---

### Prototypes

```
XVT_Pnt  
GetDimVect() const
```

### Return Value

A point whose X value is the width of the rectangle and whose Y value is the height of the rectangle.

---

## XVT\_Rct::GetTopLeft

RETRIEVE THE UPPER-LEFT CORNER OF A RECTANGLE

---

### Prototypes

```
XVT_Pnt  
GetTopLeft() const
```

### Return Value

The point at the upper left corner of the rectangle.

---

## XVT\_Rct::GetTopRight

RETRIEVE THE UPPER-RIGHT CORNER OF A RECTANGLE

---

### Prototypes

```
XVT_Pnt  
GetTopRight() const
```

### Return Value

The point at the upper right corner of the rectangle.

---

## XVT\_Rct::Height

RETRIEVE A RECTANGLE'S HEIGHT

---

### Prototypes

```
short  
Height() const
```

### Return Value

The rectangle's height.

---

## XVT\_Rct::Intersect

COMPUTE THE INTERSECTION OF TWO RECTANGLES

---

### Prototypes

```
BOOLEAN  
Intersect(  
    XVT_Rct&          boundary,  
    XVT_Rct*          intersection ) const
```

### Parameters

**boundary**  
The rectangle to check against this one for intersection.

**intersection**  
A pointer to a rectangle in which to store the intersection rectangle if there is one.



**Return Value**

A flag that is TRUE if the rectangles intersect, FALSE if they are disjoint.

**Description**

Compute the intersection of two rectangles.

**Equivalent C Function**

rect\_intersect()

---

## XVT\_Rct::IsEmpty

DETERMINE IF A RECTANGLE IS EMPTY

---

**Prototypes**

BOOLEAN  
IsEmpty() const

**Return Value**

A flag that is TRUE if the rectangle is empty, FALSE if not.

**Equivalent C Function**

is\_rect\_empty()

---

## XVT\_Rct::Normalize

NORMALIZE A RECTANGLE

---

**Prototypes**

XVT\_Rct  
Normalize() const

**Return Value**

A rectangle with the same dimensions but with its upper-left corner being (0,0).

---

## XVT\_Rct::SetBottomLeft

SET THE LOWER-LEFT CORNER OF A RECTANGLE

---

### Prototypes

```
void  
SetBottomLeft(  
    XVT_Pnt                point )
```

### Parameters

point  
The new point at the lower-left corner of the rectangle.

### Description

Sets the lower-left corner of the rectangle.

---

## XVT\_Rct::SetBottomRight

SET THE LOWER-RIGHT CORNER OF A RECTANGLE

---

### Prototypes

```
void  
SetBottomRight(  
    XVT_Pnt                point)
```

### Parameters

point  
The new point at the lower-right corner of the rectangle.

### Description

Sets the lower-right corner of the rectangle.

---

## XVT\_Rct::SetTopLeft

SET THE UPPER-LEFT CORNER OF A RECTANGLE

---

### Prototypes

```
void  
SetTopLeft(  
    XVT_Pnt                point )
```

**Parameters**

point  
The new point at the upper-left corner of the rectangle.

**Description**

Sets the upper-left corner of the rectangle.

---

## XVT\_Rct::SetTopRight

SET THE UPPER-RIGHT CORNER OF A RECTANGLE

---

**Prototypes**

```
XVT_Pnt  
SetTopRight(  
    XVT_Pnt                point )
```

**Parameters**

point  
The new point at the upper-right corner of the rectangle.

**Description**

Sets the upper-right corner of the rectangle.

---

## XVT\_Rct::TransToGlobal

TRANSLATE A POINT RELATIVE TO A RECTANGLE

---

**Prototypes**

```
XVT_Pnt  
TransToGlobal(  
    XVT_Pnt                point ) const
```

**Parameters**

point  
A point relative to the rectangle.

**Return Value**

A point in the same coordinate system as the rectangle.

**Description**

Translates a point specified relative to the rectangle to the global coordinate system.

---

## XVT\_Rct::TransToLocal

TRANSLATE A POINT RELATIVE TO THE ORIGIN

---

### Prototypes

```
XVT_Pnt  
TransToLocal(  
    XVT_Pnt                point ) const
```

### Parameters

**point**  
A point in the same coordinate system as the rectangle.

### Return Value

The same point relative to the rectangle's origin.

### Description

Translates a point from the rectangle's coordinate system to the coordinate system whose origin is the same as the rectangle's upper-left corner.

---

## XVT\_Rct::Width

RETRIEVE A RECTANGLE'S WIDTH

---

### Prototypes

```
short  
Width() const
```

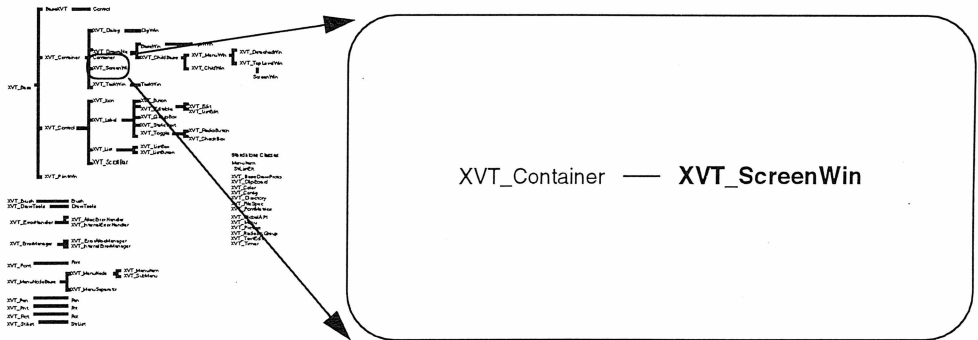
### Return Value

The rectangle's width.

## Implementation Members

```
ConvertTo  
ConvertFrom  
MakeCorners  
TopLeft  
BottomRight
```

# XVT\_ScreenWin



## Overview

Header File	screen.h
Source File	screen.cc
Superclass	XVT_Container
Subclasses	
Usage	Concrete

The XVT\_ScreenWin class defines the interface to the screen window. There is only one instance of this class and it is created by the task window and kept in the static field XVT\_Base::\_ScreenWin.

The screen window represents the physical display screen. It receives no events. Its boundaries and dimensions reflect the pixel extent of the physical screen.

## Constructors

```
XVT_ScreenWin()
virtual ~XVT_ScreenWin()
```

## Member Functions

---

### XVT\_ScreenWin::GetFirstWin

RETRIEVE THE FIRST WINDOW IN THE LIST OF CHILD WINDOWS AND DIALOGS

---

#### Prototypes

```
XVT_Container*  
GetFirstWin()
```

#### Return Value

A pointer to the first window in the list of detached windows and dialogs maintained by this window.

#### Description

Retrieves the first window in the list of detached windows and dialogs and resets the traversal context used by `GetNextWin` to the beginning of the window list.

You can retrieve all detached windows and dialogs by calling `GetFirstWin` and then calling `GetNextWin` repeatedly until it returns `NULL`.

---

### XVT\_ScreenWin::GetNextWin

RETRIEVE THE NEXT WINDOW IN THE LIST OF CHILD WINDOWS AND DIALOGS

---

#### Prototypes

```
XVT_Container*  
GetNextWin()
```

#### Return Value

A pointer to the next window or dialog relative to the current traversal context, or `NULL` if the end of the list of detached windows has been reached.

## Description

Retrieves the next detached window or dialog and increments the context.

You can retrieve all detached windows and dialogs by calling `GetFirstWin` and then calling `GetNextWin` repeatedly until it returns `NULL`.

## Equivalent C Function

```
list_windows()
```

---

# XVT\_ScreenWin::GetWinCount

RETRIEVE THE NUMBER OF CHILD WINDOWS AND DIALOGS

---

## Prototypes

```
long  
GetWinCount()
```

## Return Value

The number of detached windows and dialogs contained by this window.

## Implementation Members

```
Install  
RemoveWin  
Created
```

## Inherited Member Functions

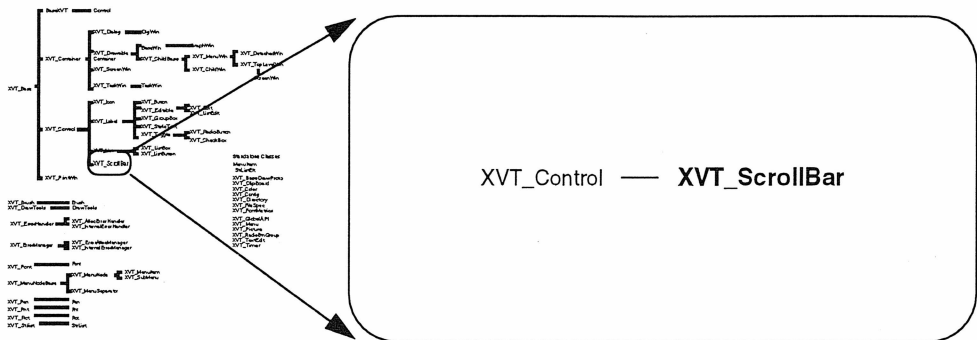
### From XVT\_Base

```
page 11    virtual BaseWin* CastToBaseWin()  
page 10    virtual DlgWin* CastToDlgWin()  
page 10    virtual ScreenWin* CastToScreenWin11()  
page 10    virtual TaskWin* CastToTaskWin11()  
page 11    virtual XVT_Button *CastToButton()  
page 11    virtual XVT_CheckBox *CastToCheckBox()
```

*page 11*     virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*     virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*     virtual XVT\_Dialog \*CastToDialog()  
*page 11*     virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*     virtual XVT\_Edit \*CastToEdit()  
*page 11*     virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*     virtual XVT\_Icon \*CastToIcon()  
*page 11*     virtual XVT\_ListBox \*CastToListBox()  
*page 11*     virtual XVT\_ListButton \*CastToListButton()  
*page 11*     virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*     virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*     virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*     virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*     virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*     virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*     virtual XVT\_StaticText \*CastToStaticText()  
*page 11*     virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*     virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*     virtual XVT\_Rct GetInnerRect()  
*page 13*     virtual XVT\_Rct GetOuterRect()



# XVT\_ScrollBar



# Overview

<b>Header File</b>	<code>scroll.h</code>
<b>Source File</b>	<code>scroll.cc</code>
<b>Superclass</b>	<code>XVT_Control</code>
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The `XVT_ScrollBar` class specifies the interface to scrollbar controls.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

Horizontal or vertical scrollbars are controls that allow the application user to manipulate an integer value in a range. The current value is represented by a “thumb,” which can be dragged with the mouse. In addition to the thumb, there are typically ways to increment or decrement the value in small steps, which we call “line-

up” and “line-down,” and large steps, which we call “page-up” and “page-down.” While the actual meaning of those modifications is defined by the application, the convention is that they work as you might expect for text being scrolled in a window.

## Constructors

```
XVT_ScrollBar( XVT_Dialog* parent, long cid,
               long min = 0L, long max = 100L)
XVT_ScrollBar( XVT_DrawableContainer* parent,
               long cid, long min = 0L, long max = 100L)
    Create a scrollbar in a window or dialog with the given range.
virtual ~XVT_ScrollBar()
```

## Member Functions

---

### XVT\_ScrollBar::e\_action

---

RECEIVE NOTIFICATION OF SCROLLBAR ACTIVITY

---

#### Prototypes

```
virtual void
e_action(
    SCROLL_CONTROL    what,
    short              pos )
```

#### Parameters

**what**  
What part of the scrollbar was manipulated.

**pos**  
The scrollbar thumb position.

#### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to scrollbar manipulation.

Each mouse click on the scrollbar generates a call to this member function. If the user holds the mouse button down, a series of events will occur.

The appearance of the scroll bar will not change unless your application explicitly modifies it, typically with `SetScrollPosition`.

The `SCROLL_CONTROL` enumeration includes the following values:

`SC_NONE`

No activity. Ignore.

`SC_LINE_UP`

Request to move one line up.

`SC_LINE_DOWN`

Request to move one line down.

`SC_PAGE_UP`

Request to move one page up.

`SC_PAGE_DOWN`

Request to move one page down.

`SC_THUMB`

The thumb has been released at the position given by `pos`.

`SC_THUMBTRACK`

The thumb has been dragged to the position given by `pos` but has not yet been released. You can safely ignore this type of event if you do not want to scroll as the user drags the thumb.

The interpretation of line and page is entirely up to the application. However, application users expect native look-and-feel guidelines to be followed.

---

## XVT\_ScrollBar::GetScrollPosition

RETRIEVE A SCROLLBAR'S THUMB POSITION

---

### Prototypes

```
long  
GetScrollPosition() const
```

### Return Value

The current position of the thumb.

### Equivalent C Function

```
get_scroll_range()
```

---

## XVT\_ScrollBar::GetScrollProportion

RETRIEVE A SCROLLBAR'S THUMB PROPORTION

---

### Prototypes

```
long  
GetScrollProportion() const
```

### Return Value

The current thumb proportion.

### Equivalent C Function

```
get_scroll_proportion()
```

---

## XVT\_ScrollBar::GetScrollRange

RETRIEVE A SCROLLBAR'S RANGE

---

### Prototypes

```
void  
GetScrollRange(  
    long* min,  
    long* max ) const
```

### Parameters

```
min  
    A pointer to a variable that receives the minimum.  
max  
    A pointer to a variable that receives the maximum.
```

### Equivalent C Function

```
get_scroll_range()
```

---

## XVT\_ScrollBar::Init

INITIALIZE A SCROLLBAR

---

### Prototypes

```
virtual BOOLEAN  
Init(  
    XVT_Rct                boundary,  
    long                   flags = 0L )
```

### Parameters

**boundary**  
The bounding rectangle for the scrollbar. If the rectangle is wider than it is tall, the scrollbar will be horizontal, if not, vertical. A boundary with a zero dimension indicates that the system default value for that dimension (width or height) is to be used.

**flags**  
Attribute flags.

### Return Value

TRUE if the scrollbar was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by the scrollbar. Recovery can be attempted by disposing of the new control, closing another control, and retrying the creation of the control.

### Description

Creates the native scrollbar if it does not already exist. If the scrollbar is in a window or dialog that was created from resources, the underlying control already exists and the `XVT_Control::Init` member function should be used instead.

`Init( boundary, flags = 0L )`  
Creates a scrollbar with the given boundary and attribute flags.

### Equivalent C Function

```
create_control()  
create_def_control()
```

---

## XVT\_ScrollBar::SetScrollPosition

SET A SCROLLBAR'S THUMB POSITION

---

### Prototypes

```
void  
SetScrollPosition(  
    long                pos )
```

### Parameters

**pos**  
The new thumb position. If the new position is outside of the current range, it will be constrained to be in that range.

### Description

Sets a scrollbar's thumb position.

This is the only way to change the thumb position. It will not happen automatically.

### Implementation Notes

XVT/CH  
Because possible thumb positions are limited to a very small number of characters, it is usual to have many values for pos map into identical thumb positions.

### Equivalent C Function

```
set_scroll_pos()
```

---

## XVT\_ScrollBar::SetScrollProportion

SET THE SIZE OF A SCROLLBAR'S THUMB

---

### Prototypes

```
void  
SetScrollProportion(  
    long                prop )
```

### Parameters

**prop**  
A value between 0 and the extent of the current scrollbar range (max - min). If the value is not in this range, it will be constrained.

## Description

This function sets the thumb proportion of a scrollbar. Conceptually, the thumb proportion is the part of a document or drawing that is visible in the viewable area, compared to the total size of the document. The word “proportion” is a bit misleading; it should be thought of as a sub-range.

The usable range of the scrollbar decreases by the size of the scroll proportion. In general, if the range is set to (*range\_min*, *range\_max*), and the proportion is set to *proportion*, then the range of possible scrollbar positions will be from *range\_min* to (*range\_max* — *proportion*).

For example, if the range were set to (–100, 100), and the proportion were set to 50, then the range of possible scrollbar positions would be (–100, 50).

## Implementation Notes

XVT/CH, XVT/Mac

Proportional thumbs are not supported.

## Equivalent C Function

set\_scroll\_proportion()

---

# XVT\_ScrollBar::SetScrollRange

SET A SCROLLBAR'S RANGE

---

## Prototypes

```
void
SetScrollRange(
    long                min,
    long                max,
    long                pos )
```

## Parameters

*min*

The lower bound of the new scrollbar range.

*max*

The upper bound of the new scrollbar range.

*pos*

The new thumb position. If the new position is outside of the new range, it will be constrained to be in that range.

**Description**

Sets a scrollbar's range and a new thumb position in that range.

**Equivalent C Function**

set\_scroll\_range()

**Implementation Members**

```
virtual BOOLEAN Init( XVT_ControlEntry* ctl_def )
ScrollProtocol
InitUpperLimit
InitLowerLimit
```

**Inherited Member Functions****From XVT\_Control**

```
page 92    virtual void Close()
page 93    virtual void e_create()
page 93    virtual void e_destroy()
page 94    virtual long e_user( long id, void *data )
page 95    BOOLEAN GetEnabledState()
page 95    long GetID( void )
page 95    XVT_Base *GetParent( void )
page 96    BOOLEAN GetVisibleState()
page 96    void Init()
page 96    void MakeFront()
page 97    void SetEnabledState( BOOLEAN state )
page 98    void SetInnerRect( XVT_Rct boundary )
page 98    void SetVisibleState( BOOLEAN state )
```

**From XVT\_Base**

```
page 11    virtual BaseWin* CastToBaseWin()
page 10    virtual DlgWin* CastToDlgWin()
page 10    virtual ScreenWin* CastToScreenWin11()
```



<i>page 10</i>	<code>virtual TaskWin* CastToTaskWin11()</code>
<i>page 11</i>	<code>virtual XVT_Button *CastToButton()</code>
<i>page 11</i>	<code>virtual XVT_CheckBox *CastToCheckBox()</code>
<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer*CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>



## Inherited Member Functions

### From XVT\_Label

- page 239*    void GetTitle( char\* str, unsigned long\* len )
- page 239*    virtual BOOLEAN Init( XVT\_Rct boundary, long = 0L, char \*  
                              = NULL )
- page 240*    void SetTitle( char\* str )

### From XVT\_Control

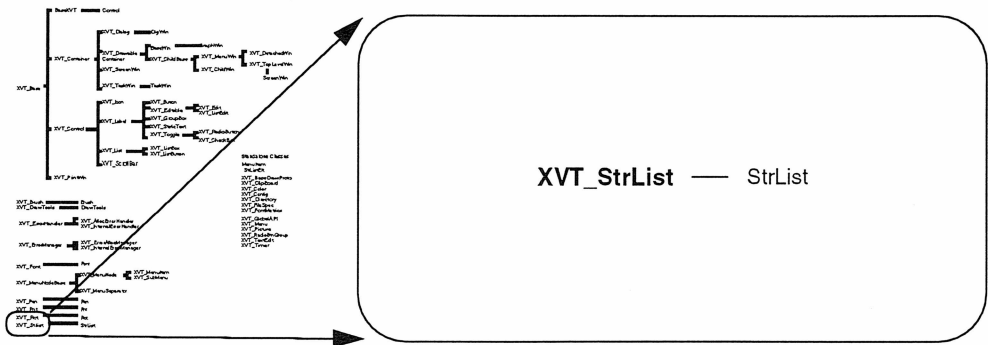
- page 92*    virtual void Close()
- page 93*    virtual void e\_create()
- page 93*    virtual void e\_destroy()
- page 94*    virtual long e\_user( long id, void \*data )
- page 95*    BOOLEAN GetEnabledState()
- page 95*    long GetID( void )
- page 95*    XVT\_Base \*GetParent( void )
- page 96*    BOOLEAN GetVisibleState()
- page 96*    void Init()
- page 96*    void MakeFront()
- page 97*    void SetEnabledState( BOOLEAN state )
- page 98*    void SetInnerRect( XVT\_Rct boundary )
- page 98*    void SetVisibleState( BOOLEAN state )

### From XVT\_Base

- page 11*    virtual BaseWin\* CastToBaseWin()
- page 10*    virtual DlgWin\* CastToDlgWin()
- page 10*    virtual ScreenWin\* CastToScreenWin11()
- page 10*    virtual TaskWin\* CastToTaskWin11()
- page 11*    virtual XVT\_Button \*CastToButton()
- page 11*    virtual XVT\_CheckBox \*CastToCheckBox()
- page 11*    virtual XVT\_ChildWin \*CastToChildWin()

<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer*CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>

# XVT\_StrList



## Overview

<b>Header File</b>	strlist.h
<b>Source File</b>	strlist.cc
<b>Superclass</b>	
<b>Subclasses</b>	StrList
<b>Usage</b>	Concrete

The XVT\_StrList class specifies the interface to general-purpose lists of strings. Each entry in a list consists of a string and a long data element.

XVT++ uses string lists wherever there is a need to represent a list of strings.

## Constructors

```
XVT_StrList( SLIST list = NULL )
XVT_StrList( const XVT_StrList& list )
~XVT_StrList()
```

## Operators

```
XVT_StrList& operator=( const XVT_StrList& list )
BOOLEAN operator==( const XVT_StrList& list )
BOOLEAN operator != ( const XVT_StrList& list )
```

## Member Functions

---

### XVT\_StrList::Add

ADD AN ITEM OR ITEMS TO A STRING LIST

---

#### Prototypes

```
void
Add(
    long                element,
    const char*         str,
    long                data = 0L )

void
Add(
    const char*         str,
    long                data = 0L )

void
Add(
    XVT_StrList*        sl )
```

#### Parameters

**element**  
The index of the element before which the element or elements are to be added. The first element is index 0. The last element may be indicated by -1.

**str**  
The string portion of the element to be added.

**data**  
The data portion of the element to be added.

**sl**  
The string list to be added to this.

**Description**

Adds an item or items to a string list.

Add( element, str, data )

Add a single element to this string list.

Add( element, sl )

Add all the elements in the string list, sl, to this string list.

Add( str, data )

Add a single element to the end of this string list.

Add( sl )

Add all the elements in the string list, sl, to the end of this string list.

**Equivalent C Function**

slist\_add()

---

**XVT\_StrList::AddSorted**

ADD AN ELEMENT TO A STRING LIST IN ORDER

---

**Prototypes**

```
void
AddSorted(
    const char*    str,
    long           data = 0L,
    BOOLEAN        unique = FALSE,
    BOOLEAN        case_sensitive = FALSE )
```

**Parameters**

str

The string portion of the element to add.

data

The data portion of the element to add.

unique

A flag that is TRUE if duplicate elements are not to be added to the string list, FALSE if they are.

case\_sensitive

A flag that is TRUE if element comparisons are to be case-sensitive, FALSE if they are to ignore case.

**Description**

Adds an element to a string list in lexicographic order.

**Equivalent C Function**

`slist_add_sorted()`

---

## XVT\_StrList::Count

RETRIEVE THE NUMBER OF ELEMENTS IN A STRING LIST

---

**Prototypes**

```
long  
Count()
```

**Return Value**

The number of elements in a string list.

**Equivalent C Function**

`slist_count()`

---

## XVT\_StrList::Debug

APPEND A DUMP OF A STRING LIST TO THE DEBUG FILE

---

**Prototypes**

```
void  
Debug()
```

**Description**

Appends a dump of a string list to the debug file.

**Equivalent C Function**

`slist_dbg()`



---

## XVT\_StrList::GetElement

RETRIEVE AN ELEMENT FROM A STRING LIST

---

### Prototypes

```
void  
GetElement(  
    long                index,  
    const char**        str,  
    long*               data ) const
```

### Parameters

**index**  
The index of the element to retrieve. A -1 indicates that the last item in the list is to be retrieved.

**str**  
Storage for a string pointer that will be set to point to the string in the element.

**data**  
Storage to receive the data associated with the element.

### Return Value

TRUE if the element was found, FALSE if index was out of range.

### Description

Retrieves an element from a string list. If index is invalid, an XVT++ internal error is generated.

### Equivalent C Function

slist\_elt()

---

## XVT\_StrList::GetFirst

START A TRAVERSAL OF A STRING LIST

---

### Prototypes

```
void  
GetFirst(  
    const char**        str,  
    long*               data )
```

**Parameters**

`str`  
Storage for a string pointer that will be set to point to the string in the element.

`data`  
Storage to receive the data associated with the element.

**Return Value**

TRUE if the element was found, FALSE if index was out of range.

**Description**

Retrieves the first element and sets the traversal context such that subsequent calls to `Next` will retrieve the second through the `N`th elements. If the list is empty, sets `str` to NULL.

**Equivalent C Function**

`slist_first()`

---

## XVT\_StrList::GetNext

RETRIEVE THE NEXT ELEMENT IN A STRING LIST

---

**Prototypes**

```
void  
GetNext(  
    const char**    str,  
    long*           data )
```

**Parameters**

`str`  
Storage for a string pointer that will be set to point to the string in the element.

`data`  
Storage to receive the data associated with the element.

**Return Value**

A flag that is TRUE if there was an element, FALSE if the end of the list was reached.

**Description**

Retrieves subsequent elements of a string list. If the end of the list has been reached, sets `str` to `NULL`.

**Equivalent C Function**

`slist_next()`

---

## XVT\_StrList::Remove

REMOVE AN ELEMENT FROM A STRING LIST

---

**Prototypes**

```
void  
Remove(  
    long                index )
```

**Parameters**

`index`  
The index of the element to remove. A `-1` indicates that the last item in the list is to be removed. Out of range indexes generate an XVT++ internal error.

**Description**

Removes an element from a string list.

**Equivalent C Function**

`slist_rem()`

## Implementation Members

```
GetList  
SetList  
List  
CurrentElt  
CopyList  
FindElement  
IsValid
```



## Constructors

```
XVT_SubMenu(  
    XVT_Menu* child,  
    MENU_TAG tag = 0,  
    BOOLEAN enabled = TRUE,  
    const char* text = NULL,  
    short mkey = 0 )  
XVT_SubMenu( XVT_SubMenu& submenu )  
~XVT_SubMenu()
```

## Member Functions

---

### XVT\_SubMenu::GetSubMenuPtr

RETRIEVE THE SUBMENU POINTER

---

#### Prototypes

```
XVT_Menu*  
GetSubMenuPtr() const
```

#### Return Value

A pointer to the menu that is the actual submenu.

---

### XVT\_SubMenu::SetSubMenuPtr

SET THE SUBMENU POINTER

---

#### Prototypes

```
void  
SetSubMenuPtr(  
    XVT_Menu* submenu )
```

#### Parameters

submenu  
The submenu.

#### Description

Sets the reference to the submenu. There is nothing special about the referenced menu object other than that it is referenced by a submenu.

## Implementation Members

ConvertTo  
XVT\_SubMenu( MENU\_ITEM\* mip )  
SetOwner  
SubMenuPtr

## Inherited Member Functions

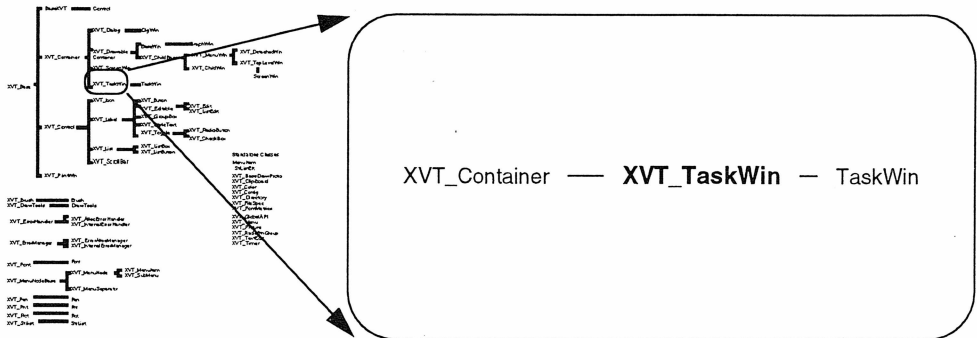
### From XVT\_MenuNode

*page 278*    `BOOLEAN GetEnabledState()`  
*page 278*    `short GetMKey()`  
*page 278*    `void GetTitle( char *buffer, long len )`  
*page 279*    `void SetEnabledState( BOOLEAN state )`  
*page 279*    `void SetTitle( char *str )`

### From XVT\_MenuNodeBase

*page 281*    `virtual XVT_MenuItem *CastToMenuItem()`  
*page 281*    `virtual XVT_MenuNode *CastToMenuNode()`  
*page 281*    `virtual XVT_MenuSeparator *CastToMenuSeparator()`  
*page 281*    `virtual XVT_SubMenu *CastToSubMenu()`  
*page 282*    `XVT_Menu *GetParent()`

# XVT\_TaskWin



# Overview

<b>Header File</b>	taskwin.h
<b>Source File</b>	taskwin.cc
<b>Superclass</b>	XVT_Container
<b>Subclasses</b>	TaskWin
<b>Usage</b>	Abstract

The `XVT_TaskWin` class specifies the interface to the task window.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

## Example

Every application must have a task window subclass that overrides at least the `e_create` member function. The following is an example of a simple task window subclass:

```
class MyTaskWin : public XVT_TaskWin
{
    void e_create();
    void e_close();
}
.
.
.
```

We create a subclass for the quit menu item, which simply closes the task window.

```
class QuitMenuItem : public XVT_MenuItem
{
public:
    QuitMenuItem( XVT_TaskWin *tw ) :
        XVT_MenuItem( M_FILE_QUIT ),
        theTaskWindow(tw);
    void e_action( BOOLEAN, BOOLEAN );

private:
    XVT_TaskWin* theTaskWindow;
};

void
QuitMenuItem::e_action( BOOLEAN, BOOLEAN )
{
    theTaskWindow->Close();
}
```

When the task window is created, we replace the default quit menu item with our subclass.

```
MyTaskWin::e_create()
{
    XVT_MenuItem* theQuitItem =
        new QuitMenuItem( this );
    .
    .
    .
}
```



```
MyTaskWin::e_close()
{
    Close();
}
```

The task window's `e_create()` function should also create whatever initial windows or dialogs your application requires.

The main function just instantiates our class window subclass:

```
int
main( int argc, char *argv[] )
{
    MyTaskWin *task_win = new MyTaskWin;
    XVT_Config config(
        MENU_BAR_RID,
        0,
        "MyApp",
        "MyApp",
        "MyApp" );

    task_win->Init( argc, argv, 0L, config );

    return 0;
}
```

## Constructors

```
XVT_TaskWin()
virtual ~XVT_TaskWin()
```

## Member Variables

---

### XVT\_TaskWin::Menu

A POINTER TO THE WINDOW'S MENU

---

#### Declaration

```
protected:

XVT_Menu* Menu;
```

**Description**

A pointer to the window's menu. Typically, you use this member when replacing default menu items with you own in a window's `e_create` implementation.

**Member Functions**

The following functions work exactly as for `XVT_DrawableContainer`:

*page 136*    `virtual void e_size( XVT_Rct boundary )`  
*page 137*    `virtual void e_timer( XVT_Timer* timer )`  
*page 139*    `virtual long e_user( long id, void* data )`  
*page 141*    `EVENT_MASK GetEventMask() const`  
*page 142*    `XVT_ChildBase* GetFirstWin()`  
*page 143*    `XVT_ChildBase* GetNextWin()`  
*page 143*    `long GetWinCount() const`  
*page 146*    `void SetEventMask( EVENT_MASK ask )`

The following functions work exactly as for `XVT_MenuWin`:

*page 287*    `virtual void e_font( XVT_Font font, FONT_PART part )`  
*page 287*    `XVT_Menu* GetMenu()`  
*page 288*    `BOOLEAN GetTitle( char* buffer, unsigned long* len ) const`  
*page 289*    `void SetFontMenu( XVT_Font font )`  
*page 290*    `void SetMenu( XVT_Menu* menu )`  
*page 291*    `void SetTitle( const char* str )`

---

**XVT\_TaskWin::Close**

SCHEDULE AN APPLICATION'S TERMINATION

---

**Prototypes**

```
void
Close()
```

**Description**

Schedules an application's termination. At some time after making this call, the `e_destroy` function is called to indicate that the application is terminating.

**Equivalent C Function**

```
close_window()
xvt_terminate()
```

---

**XVT\_TaskWin::e\_close**

RECEIVE NOTIFICATION OF A CLOSE REQUEST

---

**Prototypes**

```
virtual void
e_close()
```

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to close requests.

Typically this function is called in response to the user operating the close control on the task window. Your implementation should save and close all documents and then schedule the termination of the application by calling `Close`.

---

**XVT\_TaskWin::e\_create**

RECEIVE NOTIFICATION OF APPLICATION CREATION

---

**Prototypes**

```
virtual void
e_create()
```

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to application creation.

Your implementation should initialize your application data structures and create whatever initial windows and dialogs are required by your application.

---

## XVT\_TaskWin::e\_destroy

RECEIVE NOTIFICATION OF IMMINENT APPLICATION EXIT

---

### Prototypes

```
virtual void  
e_destroy()
```

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to application termination.

This is the last time your application receives control. At this point, all of the XVT++ interface is inoperable. You cannot (and need not) create or destroy windows or controls. You should, however, release any resources (locks, etc.) allocated by your application.

---

## XVT\_TaskWin::e\_quit

RECEIVE NOTIFICATION OF A QUIT REQUEST

---

### Prototypes

```
virtual void  
e_quit(  
    BOOLEAN                query )
```

### Parameters

query

A flag that is TRUE if the application should prepare itself to quit, and FALSE if the application should quit immediately by closing the task window.

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to quit requests.

This function is called when the user or the system wants the application to quit, or at least to consider quitting.

If your application has a Quit or Exit item on a menu (File menu, usually), this function is not called when the user chooses that item—a call to the appropriate `e_action` member is made as usual. This function is reserved for those cases where the native GUI system has the ability to tell applications that the system is performing a system-wide shutdown; it is not an event that the user can directly generate.

There are two things that should be done in an `e_quit` implementation:

query is TRUE:

The application should not actually quit, but should prepare to quit. Usually, if there are any unsaved documents, the application will have to query the user about each via a dialog box containing three buttons: Save, Discard, and Cancel. The application should do the following in each case:

*Save*

Save the document and, if that is successful, close the window and go on to the next document's save dialog.

*Discard*

Don't save the document, but just close the window and go on to the next document's save dialog.

*Cancel*

Return from the event handler without showing any more save dialogs. XVT will understand that quitting is not okay.

After the user has been queried about every unsaved document, and has not clicked the Cancel button for any of them, the application should call `QuitOK` and then return from this function to tell XVT++ that the application is willing to quit. However, it shouldn't actually quit because other applications may have to be queried also, and one of them might decline to quit.

query is FALSE:

The application was previously queried with an `e_quit(TRUE)`, and it has already determined that quitting is okay. It should immediately call the task window's `Close`. No documents have to be saved because they were taken care of earlier.

Remember that `e_quit` is different from `e_close`. The `e_quit` event is not called when the user attempts to close the task window, or any other window or dialog. In those cases, `e_close` is called.

## Implementation Notes

This call is generated only by XVT/Win and XVT/PM.

---

# XVT\_TaskWin::Init

INITIALIZE THE TASK WINDOW

---

## Prototypes

```
virtual void
Init(
    int          argc,
    char*        argv[],
    unsigned long flags,
    XVT_Config   config )
```

## Parameters

**argc**  
The value of `argc` as passed into `main`.

**argv**  
The value of `argv` as passed into `main`.

**flags**  
Attribute flags governing the task window. This parameter is currently unused.

**config**  
The `config` structure that defines the appearance of the task window and provides application-wide parameters to XVT++.

## Return Value

This call never returns.

## Description

This is the entry point into XVT++. Your entire application will run below this function call.

## Equivalent C Function

```
xvt_system()
```

---

## XVT\_TaskWin::QuitOK

INDICATE THAT THE APPLICATION CAN QUIT

---

### Prototypes

```
void  
QuitOK()
```

### Return Value

None.

### Description

This member function is used on systems where the native GUI has the ability to tell applications that the system is performing a system-wide shutdown. Its purpose is to tell XVT that the application is willing to quit. Your application will call `QuitOK` from within the `e_quit` member function of the task window, typically after it has given the user the chance to save work and confirm that it is okay to quit.

Here is an example:

```
class MyTaskWin : public XVT_TaskWin  
{  
public:  
    void e_quit( BOOLEAN query );  
  
protected:  
    BOOLEAN SaveAllDocuments();  
  
};  
  
void MyTaskWin::e_quit( BOOLEAN query )  
{  
    if (query)  
    {  
        if (SaveAllDocuments())  
            QuitOK();  
    }  
    else  
        Close();  
}
```

## Implementation Members

Install  
 RemoveWin  
 GetMenuNode  
 Created  
 AllocErrorHandler  
 InternalErrorHandler  
 TitleProtocol  
 CloseProtocol  
 MenuBarID  
 CommandEvent

## Inherited Member Functions

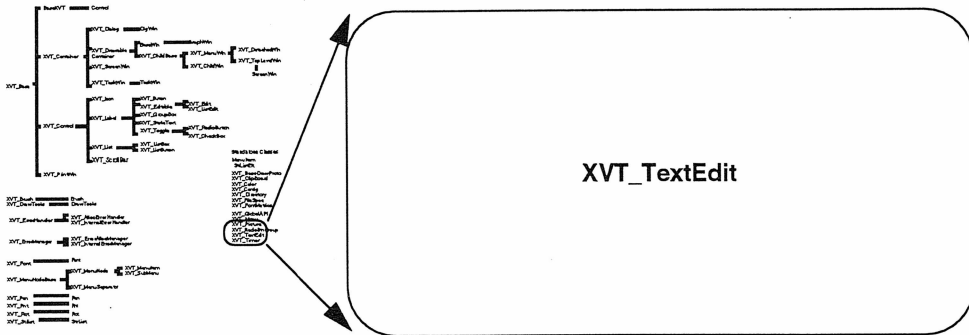
### From XVT\_Base

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()



<i>page 11</i>	virtual XVT_PrintWin *CastToPrintWin()
<i>page 11</i>	virtual XVT_RadioButton *CastToRadioButton()
<i>page 11</i>	virtual XVT_ScreenWin *CastToScreenWin()
<i>page 11</i>	virtual XVT_ScrollBar *CastToScrollBar()
<i>page 11</i>	virtual XVT_StaticText *CastToStaticText()
<i>page 11</i>	virtual XVT_TaskWin *CastToTaskWin()
<i>page 11</i>	virtual XVT_TopLevelWin *CastToTopLevelWin()
<i>page 12</i>	virtual XVT_Rct GetInnerRect()
<i>page 13</i>	virtual XVT_Rct GetOuterRect()

# XVT\_TextEdit



# Overview

<b>Header File</b>	textedit.h
<b>Source File</b>	textedit.cc
<b>Superclass</b>	
<b>Subclasses</b>	
<b>Usage</b>	Concrete

A Text Edit *object* consists of text divided into *paragraphs*, each terminated with a carriage return. If word-wrapping is enabled, each paragraph can appear as one or more *lines* of text.

The length of a line and the total number of lines can be very large, so normally not all of the text appears on the screen. Only the part within the view rectangle can be seen. The application program or the user can scroll the text so that the unseen part enters the view rectangle.

Optionally, you can request the Text Edit system to draw a *border* rectangle around the view rectangle. If so, the view rectangle is inset by 4 pixels inside the border rectangle. On the bottom it may be inset by a few more pixels, because the bottom coordinate of the view

rectangle you request may be reduced so that an integral number of text lines will appear.

Note that in XVT++ text edit objects automatically create a borderless child window to contain them.

## Constructors

```
XVT_TextEdit( XVT_ChildBase* parent)
virtual ~XVT_TextEdit()
```

## Member Functions

---

### XVT\_TextEdit::Activity

DETERMINE IF THERE HAS BEEN ANY USER ACTIVITY IN A TEXT EDIT

---

#### Prototypes

```
BOOLEAN
Activity()
```

#### Return Value

A flag that is TRUE if the user has operated the text edit since the previous call to Activity, FALSE if not.

---

### XVT\_TextEdit::AddPar

ADD A NEW PARAGRAPH TO A TEXT EDIT OBJECT

---

#### Prototypes

```
BOOLEAN
AddPar(
    T_PNUM          t,
    char*           ch )
```

#### Return Value

TRUE if successful, FALSE if not.

**Parameters**

- t**      The paragraph before which to add the new paragraph. The first paragraph is 0.
- ch**     A null-terminated string that gives the contents of the new paragraph.

**Description**

Adds a new paragraph to a text edit object.

**Equivalent C Function**

tx\_add\_par()

---

## XVT\_TextEdit::Append

APPEND A STRING TO A PARAGRAPH

---

**Prototypes**

```

BOOLEAN
Append(
    T_PNUM          t,
    const char*     ch )

```

**Return Value**

TRUE if successful, FALSE if not.

**Parameters**

- t**      The paragraph upon which to append the string. The first paragraph is 0.
- ch**     The text to append to the paragraph.

**Description**

Appends a string to a paragraph.

**Equivalent C Function**

tx\_append()

---

## XVT\_TextEdit::Clear

REMOVE ALL TEXT FROM A TEXT EDIT OBJECT

---

### Prototypes

```
BOOLEAN
Clear()
```

### Return Value

TRUE if successful, FALSE if not.

### Description

Removes all text from a text edit object.

### Equivalent C Function

```
tx_clear()
```

---

## XVT\_TextEdit::Close

SCHEDULE THIS TEXT EDIT FOR DESTRUCTION

---

### Prototypes

```
void
Close()
```

### Description

Schedules this text edit for destruction.

### Equivalent C Function

```
tx_destroy()
```

---

## XVT\_TextEdit::DelPar

DELETE A PARAGRAPH

---

### Prototypes

```
BOOLEAN
DelPar(
    T_PNUM          t )
```

**Parameters**

t  
The paragraph to be deleted. The first paragraph is 0. If t is out of range the operation will be ignored.

**Return Value**

TRUE if successful, FALSE if not.

**Description**

Deletes a paragraph.

**Equivalent C Function**

tx\_del\_par()

---

## XVT\_TextEdit::DoHscroll

SCROLL A TEXT EDIT HORIZONTALLY

---

**Prototypes**

```
void  
DoHscroll(  
    long                x )
```

**Parameters**

x  
The number of pixels to scroll by.

**Description**

Scrolls a text edit horizontally.

**Equivalent C Function**

tx\_hscroll()

---

## XVT\_TextEdit::DoVscroll

SCROLL A TEXT EDIT OBJECT VERTICALLY

---

**Prototypes**

```
void  
DoVscroll(  
    long                l )
```

**Parameters**

l  
The number of lines to scroll.

**Description**

Scrolls a text edit object vertically.

**Equivalent C Function**

tx\_vscroll()

---

## XVT\_TextEdit::GetAttrib

RETRIEVE A TEXT EDIT OBJECT'S ATTRIBUTES

---

**Prototypes**

unsigned long  
GetAttrib() const

**Return Value**

The text edit object's current attributes.

**Equivalent C Function**

tx\_get\_attrib() const

---

## XVT\_TextEdit::GetBorder

RETRIEVE A TEXT EDIT'S BORDER RECTANGLE

---

**Prototypes**

XVT\_Rct  
GetBorder() const

**Return Value**

The text edit's border rectangle.

**Equivalent C Function**

tx\_get\_border()

---

## XVT\_TextEdit::GetFont

RETRIEVE A TEXT EDIT OBJECT'S FONT

---

### Prototypes

```
XVT_Font  
GetFont() const
```

### Return Value

The text edit object's current font.

### Equivalent C Function

```
tx_get_font()
```

---

## XVT\_TextEdit::GetLimit

RETRIEVE A TEXT EDIT OBJECT'S CHARACTER LIMIT

---

### Prototypes

```
long  
GetLimit() const
```

### Return Value

The maximum number of characters allowed in the text edit object.  
This number is meaningful only if TX\_ONEPAR is set.

### Equivalent C Function

```
tx_get_limit()
```

---

## XVT\_TextEdit::GetLine

GET THE CONTENTS OF A LINE

---

### Prototypes

```
BOOLEAN  
GetLine(  
    char*  
    T_PNUM,  
    T_LNUM,  
    unsigned long*  
    buffer  
    paragraph  
    line  
    len ) const
```



**Parameters**

**buffer**  
Storage to receive the line's contents.

**paragraph**  
The index of the paragraph.

**line**  
The index of the line in the paragraph.

**len**  
The length of the returned string.

**Return Value**

A flag indicating whether the user-supplied buffer was long enough to store the entire text line. This flag is FALSE if the buffer was not long enough, in which case the correct length is returned via the len parameter. Otherwise the flag is TRUE.

**Description**

Gets the contents of a line.

**Equivalent C Function**

tx\_get\_line()

---

## XVT\_TextEdit::GetMargin

GET THE TEXT EDIT OBJECT'S MARGIN

---

**Prototypes**

long  
GetMargin() const

**Return Value**

The current margin in pixels.

**Equivalent C Function**

tx\_get\_margin()

---

## XVT\_TextEdit::GetNumChars

RETRIEVE THE NUMBER OF CHARACTERS IN A TEXT EDIT LINE

---

### Prototypes

```
T_CNUM  
GetNumChars(  
    T_PNUM  
    T_LNUM  
    p,  
    l ) const
```

### Parameters

p  
The index of the paragraph.

l  
The index of the line in the paragraph.

### Return Value

The number of characters in the given line.

### Equivalent C Function

tx\_get\_num\_chars()

---

## XVT\_TextEdit::GetNumLines

RETRIEVE THE NUMBER OF LINES IN THE TEXT EDIT OBJECT

---

### Prototypes

```
T_LNUM  
GetNumLines() const
```

### Return Value

The number of lines in the text edit object.

### Equivalent C Function

tx\_get\_num\_lines()

---

## XVT\_TextEdit::GetNumParLines

RETRIEVE THE NUMBER OF LINES IN A TEXT EDIT PARAGRAPH

---

### Prototypes

```
T_LNUM  
GetNumParLines(  
    T_PNUM                t ) const
```

### Parameters

t  
The index of the paragraph.

### Return Value

The number of lines in the paragraph.

### Equivalent C Function

```
tx_get_num_par_lines()
```

---

## XVT\_TextEdit::GetNumPars

RETRIEVE THE NUMBER OF PARAGRAPHS IN A TEXT EDIT OBJECT

---

### Prototypes

```
T_PNUM  
GetNumPars() const
```

### Return Value

The number of paragraphs in the text edit object.

### Equivalent C Function

```
tx_get_num_pars()
```

---

## XVT\_TextEdit::GetOrigin

RETRIEVE THE OFFSET TO THE CURRENT VIEW RECTANGLE

---

### Prototypes

```
void
GetOrigin(
    T_PNUM*           p,
    T_LNUM*           l1,
    T_LNUM*           l2,
    T_CPOS*           cp ) const
```

### Parameters

p  
The paragraph number.

l1  
The line number relative to p.

l2  
The line number relative to the beginning of the text edit object.

cp  
The pixel offset relative to the left margin.

### Description

Retrieves various offsets describing the location of the current view rectangle.

### Equivalent C Function

```
tx_get_origin()
```

---

## XVT\_TextEdit::GetSel

RETRIEVE THE CURRENT TEXT EDIT SELECTION

---

### Prototypes

```
void
GetSel(
    T_PNUM*           p1,
    T_LNUM*           l1,
    T_CNUM*           c1,
    T_PNUM*           p2,
    T_LNUM*           l2,
    T_CNUM*           c2 ) const
```

**Parameters**

- p1  
The starting paragraph number.
- l1  
The starting line number.
- c1  
The starting character number.
- p2  
The ending paragraph number.
- l2  
The ending line number.
- c2  
The ending character number.

**Description**

Retrieves the current selection. If the current selection is empty, then it will be the case that (p1 == p2) && (l1 == l2) && (c1 == c2).

**Equivalent C Function**

tx\_get\_sel()

---

## XVT\_TextEdit::GetView

RETRIEVE THE VIEW RECTANGLE

---

**Prototypes**

XVT\_Rct  
GetView() const

**Return Value**

The view rectangle.

**Equivalent C Function**

tx\_get\_view()

## XVT\_TextEdit::Init

CREATE THE UNDERLYING TEXT EDIT OBJECT

### Prototypes

```

BOOLEAN
Init(
    XVT_Rct      boundary,
    unsigned short attrib,
    XVT_Font     font,
    short         margin,
    short         limit )

```

### Parameters

**boundary**

The bounding rectangle of the new text edit object.

**attrib**

The initial text edit attributes, a bitwise OR'd combination of flags. Valid attribute flags are:

**TX\_AUTOHSCROLL**

If set, enables automatic scrolling in the horizontal direction.

**TX\_AUTOVSCROLL**

If set, enables automatic scrolling in the vertical direction.

**TX\_BORDER**

If set, causes the text edit object to draw a rectangular border.

**TX\_ENABLECLEAR**

If set, leaves the clear item in the edit menu enabled.

**TX\_NOCOPY**

If set, disables the copy item in the edit menu.

**TX\_NOCUT**

If set, disables the cut item in the edit menu.

**TX\_NOMENU**

If set disables all interaction with the edit menu.

**TX\_NOPASTE**

If set, disables the paste item in the edit menu.

**TX\_ONEPAR**

If set, limits the text edit object to a single paragraph.

**TX\_OVERTYPE**

If set, causes new characters to overwrite old rather than inserting themselves.

**TX\_READONLY**

If set, makes the text edit object read-only. Users can scroll the text but cannot modify it.

**TX\_WRAP**

If set, causes text edit to wrap lines that will not fit in the text edit.

**TX\_VSCROLLBAR**

If set, causes the text edit to have a vertical scrollbar.

**TX\_HSCROLLBAR**

If set, causes the text edit to have a horizontal scrollbar.

**font**

The font to be used to render the text inside the text edit object.

**margin**

The right margin in pixels. This value is meaningful only if the TX\_WRAP attribute is set.

**limit**

The limit on the number of characters in the text edit object. This value is meaningful only if the TX\_ONEPAR attribute is set.

**Return Value**

TRUE if the text edit was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by windows. Recovery can be attempted by disposing of the new text edit, closing another window or text edit object, and retrying the creation of the text edit object.

**Description**

The Init member functions create the underlying text edit object.

Init( boundary, attrib, font, margin, limit )

Create a text edit object as specified by the given parameters.

**Equivalent C Function**

tx\_create()

create\_def\_tx()

---

## XVT\_TextEdit::Reset

RESET A TEXT EDIT OBJECT

---

### Prototypes

```
void  
Reset()
```

### Description

Resets a text edit object. Any selected text is unselected, the caret is positioned before the first character, the text is scrolled as far up and to the left as possible, all paragraphs are rewrapped, and an update event for the border rectangle is enqueued.

### Equivalent C Function

```
tx_reset()
```

---

## XVT\_TextEdit::Resume

RESUME SCREEN UPDATES

---

### Prototypes

```
void  
Resume()
```

### Description

Resumes screen updates. Cancels a previous call to Suspend.

### Equivalent C Function

```
tx_resume()
```

---

## XVT\_TextEdit::SetActive

MAKE THIS TEXT EDIT BE ACTIVE

---

### Prototypes

```
void  
SetActive()
```



**Description**

Makes this text edit active. The active text edit has keyboard focus. Calling this function causes whatever window currently has focus to lose it.

**Equivalent C Function**

tx\_set\_active()

---

**XVT\_TextEdit::SetAttrib**

SET A TEXT EDIT'S ATTRIBUTES

---

**Prototypes**

```
void  
SetAttrib(  
    unsigned long    attrib )
```

**Parameters**

attrib  
The new text edit attributes. See Init for details.

**Description**

Sets a text edit's attributes.

**Equivalent C Function**

tx\_set\_attrib()

---

**XVT\_TextEdit::SetBorder**

SET A TEXT EDIT OBJECT'S BORDER RECTANGLE

---

**Prototypes**

```
void  
SetBorder(  
    XVT_Rct    boundary )
```

**Parameters**

boundary  
The new border rectangle.

**Description**

Sets a text edit object's border rectangle.

**Equivalent C Function**

tx\_set\_border()

---

## XVT\_TextEdit::SetColors

SET TEXT EDIT OBJECT COLORS

---

**Prototypes**

```
void
SetColors(
    XVT_Color    text,
    XVT_Color    border,
    XVT_Color    background )
```

**Parameters**

text  
The text foreground color.

border  
The text border foreground color.

background  
The background color.

**Description**

Sets text edit object colors.

**Equivalent C Function**

tx\_set\_colors()

---

## XVT\_TextEdit::SetFont

SET A TEXT OBJECT'S FONT

---

**Prototypes**

```
void
SetFont(
    XVT_Font    f )
```

**Parameters**

f  
The new font.

**Description**

Sets a text object's font.

**Equivalent C Function**

tx\_set\_font()

---

## XVT\_TextEdit::SetLimit

SET A TEXT EDIT OBJECT'S CHARACTER LIMIT

---

**Prototypes**

```
void  
SetLimit(  
    long                l )
```

**Parameters**

l  
The new character limit.

**Description**

Sets a text edit object's character limit.

**Equivalent C Function**

tx\_set\_limit()

---

## XVT\_TextEdit::SetMargin

SET A TEXT EDIT OBJECT'S MARGIN

---

**Prototypes**

```
void  
SetMargin(  
    long                margin )
```

**Parameters**

margin  
The new margin.

**Description**

Sets a text edit object's margin.

**Equivalent C Function**

tx\_set\_margin()

---

## XVT\_TextEdit::SetPar

REPLACE A PARAGRAPH

---

**Prototypes**

```

BOOLEAN
SetPar(
    T_PNUM,          t
    const char*      ch )

```

**Parameters**

t  
The index of the paragraph to be replaced.

ch  
The contents of the replacement paragraph.

**Description**

Replaces a paragraph.

**Equivalent C Function**

tx\_set\_par()

---

## XVT\_TextEdit::SetSel

SET THE TEXT EDIT OBJECT'S SELECTION

---

**Prototypes**

```

void
SetSel(
    T_PNUM    p1,
    T_LNUM    l1,
    T_CNUM    c1,
    T_PNUM    p2,
    T_LNUM    l2,
    T_CNUM    c2)

```

**Parameters**

p1	The starting paragraph number.
l1	The starting line number.
c1	The starting character number.
p2	The ending paragraph number.
l2	The ending line number.
c2	The ending character number.

**Description**

Sets the text edit object's selection.

**Equivalent C Function**

tx\_set\_sel()

---

## XVT\_TextEdit::Suspend

SUSPEND SCREEN UPDATES

---

**Prototypes**

```
void  
Suspend()
```

**Description**

Suspends screen updates until the next call to Resume.

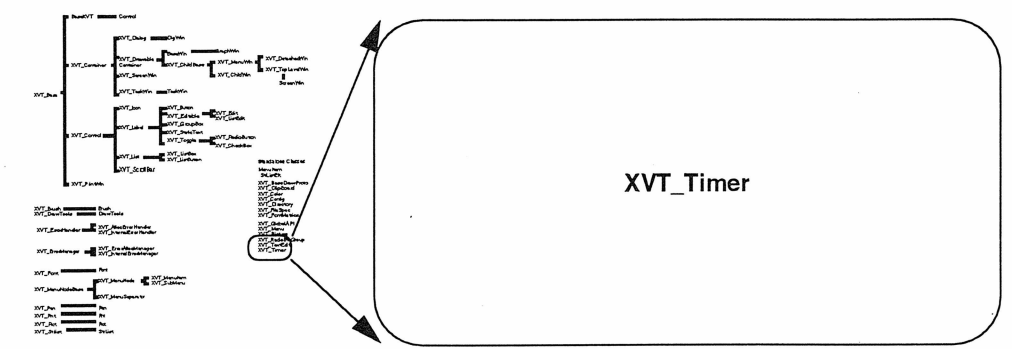
**Equivalent C Function**

tx\_suspend()

## Implementation Members

```
BOOLEAN Init( XVT_TextEditEntry* tx_def )  
GetLinesInView  
GetID  
ID  
Parent  
Enclosure
```

# XVT\_Timer



## Overview

Header File	timer.h
Source File	timer.cc
Superclass	
Subclasses	
Usage	Concrete

Instances of the XVT\_Timer class handle the creation and destruction of timers. Basically, a timer is started when an instance of this class is created and destroyed when the instance is deleted. As long as the timer object exists, the target object's e\_timer member function is called at the given interval.

XVT++ guarantees that timer intervals of one second or greater will be honored in all environments. However, timer intervals of less than one second are *not* portable.

## Example

In the action code for a button, this example starts a timer for the parent window with a 5-second interval:

```
MyButton::e_action()
{
    new XVT_Timer(
        (XVT_DrawableContainer*)GetParent(),
        5000 );
    .
    .
    .
}
```

Casting the result of `GetParent` is okay as long as the `MyButton` implementation always knows a-priori that its parent will be a subclass of `XVT_DrawableContainer`. If this is not the case, the button subclass should include constructors which retain a properly typed pointer to the parent object.

```
MyWindow::e_timer( XVT_timer *timer )
{
    delete timer;
    .
    .
    .
}
```

In the parent window's `e_timer` implementation, we delete the timer and do whatever it was that we wanted to delay for five seconds.

## Constructors

```
XVT_Timer( XVT_TaskWin* target, long interval )
XVT_Timer( XVT_DrawableContainer* target, long interval )
XVT_Timer( XVT_Dialog* target, long interval )
    Create a timer in a task window, window or dialog. The interval
    is given in milliseconds. All are essentially equivalent to
    create_timer.
~XVT_Timer()
    Dispose of a timer. Equivalent to kill_timer.
```

## Member Functions

---

### XVT\_Timer::GetInterval

RETRIEVE A TIMER'S INTERVAL

---

#### Prototypes

```
long  
GetInterval() const
```

#### Return Value

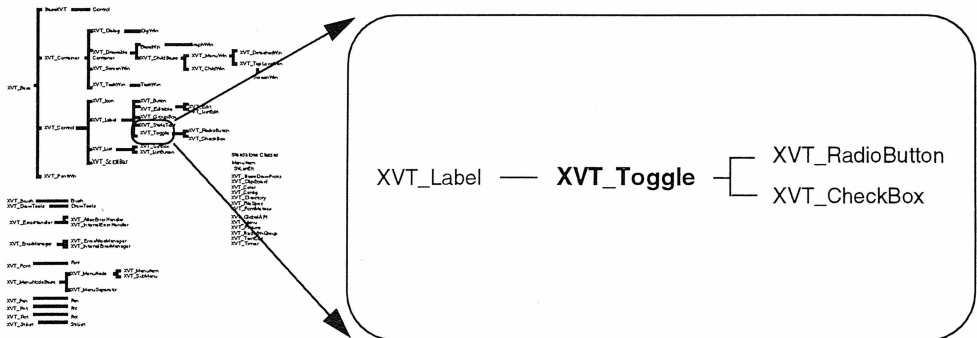
The timer's interval.

## Implementation Members

```
GetID  
GetTarget  
ID  
Interval  
Target  
DoInit
```



# XVT\_Toggle



## Overview

<b>Header File</b>	<code>toggle.h</code>
<b>Source File</b>	<code>toggle.cc</code>
<b>Superclass</b>	XVT_Label
<b>Subclasses</b>	XVT_CheckBox, XVT_RadioButton
<b>Usage</b>	Implementation

The XVT\_Toggle class defines the interface common to all two-state (toggle) controls.

## Member Functions

---

### XVT\_Toggle::e\_action

RECEIVE NOTIFICATION THAT A TOGGLE CONTROL HAS BEEN OPERATED

---

#### Prototypes

```
virtual void  
e_action()
```

#### Description

This member function is called when a toggle has been operated (toggled). The default version does nothing. Your subclass should provide a definition for this function, which does whatever you want to do when a toggle is pressed.

Typically, applications check the toggle using one of the Set\_Checked\_State member functions provided by subclasses.

---

### XVT\_Toggle::GetCheckedState

RETRIEVE THE TOGGLE CONTROL'S STATE

---

#### Prototypes

```
BOOLEAN  
GetCheckedState() const
```

#### Return Value

A flag that is TRUE if the toggle is in a non-default state, or FALSE if it is in a default state.

## Implementation Members

```
CheckProtocol
```

## Inherited Member Functions

### From XVT\_Label

- page 239*    `void GetTitle( char* str, unsigned long* len )`  
*page 239*    `virtual BOOLEAN Init( XVT_Rct boundary, long = 0L, char *  
                          = NULL )`  
*page 240*    `void SetTitle( char* str )`

### From XVT\_Control

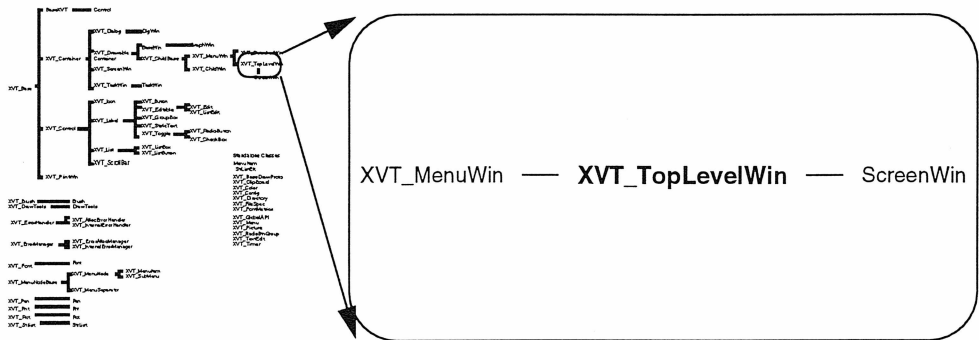
- page 92*    `virtual void Close()`  
*page 93*    `virtual void e_create()`  
*page 93*    `virtual void e_destroy()`  
*page 94*    `virtual long e_user( long id, void *data )`  
*page 95*    `BOOLEAN GetEnabledState()`  
*page 95*    `long GetID( void )`  
*page 95*    `XVT_Base *GetParent( void )`  
*page 96*    `BOOLEAN GetVisibleState()`  
*page 96*    `void Init()`  
*page 96*    `void MakeFront()`  
*page 97*    `void SetEnabledState( BOOLEAN state )`  
*page 98*    `void SetInnerRect( XVT_Rct boundary )`  
*page 98*    `void SetVisibleState( BOOLEAN state )`

### From XVT\_Base

- page 11*    `virtual BaseWin* CastToBaseWin()`  
*page 10*    `virtual DlgWin* CastToDlgWin()`  
*page 10*    `virtual ScreenWin* CastToScreenWin11()`  
*page 10*    `virtual TaskWin* CastToTaskWin11()`  
*page 11*    `virtual XVT_Button *CastToButton()`  
*page 11*    `virtual XVT_CheckBox *CastToCheckBox()`  
*page 11*    `virtual XVT_ChildWin *CastToChildWin()`

*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# XVT\_TopLevelWin



## Overview

Header File	toplevel.h
Source File	toplevel.cc
Superclass	XVT_MenuWin
Subclasses	ScreenWin
Usage	Abstract

The XVT\_TopLevelWin class specifies the interface to the class of windows that may contain controls or child windows and that are contained by the task window if the native window system has a task window.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

## Example

Most applications will have at least one top-level window subclass. The following declaration is typical:

```
class MyTopLevelWin : public XVT_TopLevelWin
{
public:
    MyTopLevelWin();
    ~MyTopLevelWin();

    void e_create();
    void e_destroy();
    void
    e_char(
        short chr,
        BOOLEAN shift,
        BOOLEAN control );

private:
    struct MyWindowData
    {
        XVT_TopLevelWin* secondaryView;
        long foo;
        // other data associated with this window...
    } Data;
};
```

The subclass overrides some event handling member functions and adds some window-specific data in a substructure. Among the window data is a pointer to another top level window, `secondaryView`, which provides a functionality related to this window.

The example supposes that this window is being created from a resource that defines not only the window but its menubar and contained controls as well. The window is created, presumably in the task window's `e_create` function, by code that looks like this:

```
{
    XVT_TopLevelWin *newWin;
    .
    .
    .

    newWin = new MyTopLevelWin;
    newWin->Init( MY_WIN_RID );
    .
    .
    .
}
```

The implementation constructors and destructors typically just initialize instance data structures. Operations on other GUI objects, creating them, or destroying them, are best handled in `e_create` and `e_destroy` because causing recursion inside a constructor or destructor could cause XVT++ to operate on objects that are not yet completely initialized.

```
MyTopLevelWin::MyTopLevelWin()
{
    Data.foo = 0;
    Data.secondaryView = (XVT_TopLevelWin*)0;
    // initialize remainder of data...
}

MyTopLevelWin::~MyTopLevelWin()
{
    delete Data...
    // deallocate memory, etc...
}
```

In the window's `e_create` method, you must create the controls contained by the window. You create a control by applying the operator new to the control subclass and then calling one of the control's `Init` methods. If the window was specified in resources, the underlying controls already exist and you can use the `Init` method with no arguments. If the control is being created at runtime, then you must use the `Init` method with parameters.

In the resource case, the native controls already exist. The `Init` method simply hooks the XVT++ control object up to the existing native control. If you attempt to use a native control that has not been hooked up to an XVT++ control, you will cause an error. In the runtime case, the native control is actually created by the `Init` call.

Menu items are similar to controls in that you must replace default menu item instances with your own menu item subclasses in order to create an operable menu. Alternatively, you could create an `XVT_Menu` structure from scratch and then associate it with the window by calling `SetMenu`:

```
class MyListbox;
class MyFileQuitMenuItem;
class MyFileOpenMenuItem;
.
.
.

void
MyTopLevelWin::e_create()
{
    // Create controls...
```

```

    {
        register XVT_Control* newControl;

        newControl = new MyListbox( this, LISTBOX_CID );
        newControl->Init();

        // and so on for the remainder of the controls
        .
        .
        .
    }

    // Create menu items...
    {
        register XVT_MenuItem* newMenuItem;

        Menu->Replace( new MyFileQuitMenuItem(...) );
        Menu->Replace( new MyFileOpenMenuItem(...) );

        // and so on for the remainder of the menu items
        .
        .
        .
    }

    // Create the associated window

    Data.secondaryView = new SecondaryViewWin(...);
    Data.secondaryView->Init( SECONDARY_VIEW_RID );
}

void
MyTopLevelWin::e_destroy()
{
    // Dispose of the associated window

    Data.secondaryView->Close();
}

void
MyTopLevelWin::e_char( short chr, BOOLEAN shift, BOOLEAN
control )
{
    // Do whatever this window does when characters
    // are received.
}

```

Note that the associated window, `Data.secondaryView`, is created and destroyed in the `e_create/e_destroy` methods and not in the



constructor/destructor methods, so as to avoid causing recursion in constructors.

The process of creating a top-level window subclass is very similar to creating detached window or dialog subclasses. With obvious modifications, you can apply this example to those classes as well.

## Constructors

**XVT\_TopLevelWin()**

Create a top level window. The actual method by which the native window will be created is determined by which Init function is called.

**virtual ~XVT\_TopLevelWin()**

Removes the top level window from the task window's list of child windows.

## Member Functions

---

### XVT\_TopLevelWin::Init

INITIALIZE THE WINDOW

---

#### Prototypes

```

BOOLEAN
Init(
    WIN_TYPE
    XVT_Rct
    const char*
    long
    long
    wtype,
    boundary,
    title,
    menu_rid,
    flags )

BOOLEAN
Init(
    long
    rid )
  
```

#### Parameters

**wtype**

The type of window to be created. It should be one of W\_DOC, W\_DBL, or W\_PLAIN.

**boundary**

The bounding rectangle (in pixels) of the window's client area. On native window-systems with a task window, the rectangle is

relative to the task window's client area. On all other native window systems, it is in screen coordinates.

**title**

The window's title. If the `wtype` is `W_DOC`, the title is set as though `SetDocTitle` had been called; otherwise, it will be set as though `SetTitle` was called.

**menu\_rid**

The resource ID for the window's menu.

**flags**

A bitwise OR'd combination of flags that control the window's attributes and decoration.

**rid**

The resource ID by means of which the window's dimensions, attributes, and contents can be located.

## Return Value

TRUE if the window was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by windows. Recovery can be attempted by disposing of the new window, closing another window, and retrying the creation of the window.

## Description

The `Init` member functions create the native window and call the window's `e_create` method. When execution returns from the `Init` call, the window is complete and ready to use. Prior to the `Init` call, the window is not usable.

`Init( wtype, boundary, title, menu_rid, flags )`

Creates only a window with the given parameters. XVT++ control objects must be created separately by the user.

`Init( rid )`

Creates a window and contained controls from a resource specification. XVT++ control objects corresponding to the controls described in the resource must be created and installed separately by the application developer. The recommended place to do this is in the window's `e_create` member function; however, the control objects may be created at any time. Events intended for controls that have no corresponding XVT++ control object will cause a run-time error.

## Equivalent C Functions

```
create_window()
create_def_window()
create_res_window()
```

## Implementation Members

```
BOOLEAN Init( XVT_WindowDef* def )
```

## Inherited Member Functions

### From XVT\_MenuWin

```
page 286 virtual void e_close()
page 287 virtual void e_font( XVT_Font font, FONT_PART part )
page 287 XVT_Menu *GetMenu()
page 288 void GetTitle( char *buffer, long len )
page 289 void SetDocTitle( char *str )
page 289 void SetFontMenu( XVT_Font font )
page 290 void SetMenu( XVT_Menu *menu )
page 291 void SetTitle( char *str )
```

### From XVT\_ChildBase

```
page 49 virtual void e_hscroll( SCROLL_CONTROL activity, short
pos )
page 49 virtual void e_vscroll( SCROLL_CONTROL activity, short
pos )
page 50 XVT_TextEdit* GetActiveTextEdit()
page 50 XVT_Pnt GetCaretPos() const
page 51 BOOLEAN GetCaretState() const
page 51 BOOLEAN GetEnabledState()
page 51 XVT_ChildBase *GetParent() const
page 52 long GetScrollPosition( SCROLL_TYPE scroll_type ) const
page 52 long GetScrollProportion( SCROLL_TYPE scroll_type ) const
```

<i>page 53</i>	<code>void GetScrollRange( SCROLL_TYPE scroll_type, long *min, long *max ) const</code>
<i>page 54</i>	<code>XVT_TextEdit* GetTextEdit( long id )</code>
<i>page 54</i>	<code>BOOLEAN GetVisibleState()</code>
<i>page 55</i>	<code>void MakeFront()</code>
<i>page 55</i>	<code>void ReleaseMouse()</code>
<i>page 56</i>	<code>void SetCaretDimensions( XVT_Pnt vector )</code>
<i>page 56</i>	<code>void SetCaretPos( XVT_Pnt point )</code>
<i>page 57</i>	<code>void SetCaretState( BOOLEAN state )</code>
<i>page 57</i>	<code>void SetCursor( CURSOR cursor )</code>
<i>page 58</i>	<code>void SetEnabledState( BOOLEAN state )</code>
<i>page 59</i>	<code>void SetScrollPosition( SCROLL_TYPE scroll_type, long position )</code>
<i>page 60</i>	<code>void SetScrollProportion( SCROLL_TYPE scroll_type, long proportion )</code>
<i>page 60</i>	<code>void SetScrollRange( SCROLL_TYPE scroll_type, long min, long max, long pos )</code>
<i>page 61</i>	<code>void SetVisibleState( BOOLEAN f )</code>
<i>page 62</i>	<code>void TrapMouse()</code>

### **From XVT\_DrawableContainer**

<i>page 129</i>	<code>void Clear()</code>
<i>page 129</i>	<code>void Clear( XVT_Color color )</code>
<i>page 129</i>	<code>void Close()</code>
<i>page 128</i>	<code>XVT_BaseDrawProto* DrawProtocol</code>
<i>page 130</i>	<code>virtual void e_char( short chr, BOOLEAN shift, BOOLEAN control)</code>
<i>page 131</i>	<code>virtual void e_create()</code>
<i>page 132</i>	<code>virtual void e_destroy()</code>
<i>page 132</i>	<code>virtual void e_focus( BOOLEAN active )</code>

- page 133*     virtual void e\_mouse\_dbl(  
                  XVT\_Pnt point,  
                  BOOLEAN shift,  
                  BOOLEAN control,  
                  short button )
- page 134*     virtual void e\_mouse\_down(  
                  XVT\_Pnt point,  
                  BOOLEAN shift,  
                  BOOLEAN control,  
                  short button )
- page 135*     virtual void e\_mouse\_move(  
                  XVT\_Pnt point,  
                  BOOLEAN shift,  
                  BOOLEAN control,  
                  short button )
- page 135*     virtual void e\_mouse\_up(  
                  XVT\_Pnt point,  
                  BOOLEAN shift,  
                  BOOLEAN control,  
                  short button )
- page 136*     virtual void e\_size( XVT\_Rct boundary )
- page 137*     virtual void e\_timer( long id )
- page 137*     virtual void e\_update( XVT\_Rct boundary )
- page 139*     virtual long e\_user( long id, void \*data )
- page 140*     XVT\_Control \*GetCtl( long cid )
- page 140*     long GetCtlCount()
- page 141*     EVENT\_MASK GetEventMask() const
- page 141*     XVT\_Control \*GetFirstCtl()
- page 142*     XVT\_ChildBase \*GetFirstWin()
- page 142*     XVT\_Control \*GetNextCtl()
- page 143*     XVT\_ChildBase \*GetNextWin()
- page 143*     long GetWinCount()
- page 144*     void Invalidate()
- page 144*     void Invalidate( XVT\_Rctregion )
- page 145*     void Scroll(  
                  XVT\_Rct boundary,  
                  long dh,  
                  long dv )

*page 146*    `void SetEventMask( EVENT_MASK ask )`

*page 148*    `void SetInnerRect( XVT_Rct r )`

### **From XVT\_Base**

*page 11*    `virtual BaseWin* CastToBaseWin()`

*page 10*    `virtual DlgWin* CastToDlgWin()`

*page 10*    `virtual ScreenWin* CastToScreenWin11()`

*page 10*    `virtual TaskWin* CastToTaskWin11()`

*page 11*    `virtual XVT_Button *CastToButton()`

*page 11*    `virtual XVT_CheckBox *CastToCheckBox()`

*page 11*    `virtual XVT_ChildWin *CastToChildWin()`

*page 11*    `virtual XVT_DetachedWin *CastToDetachedWin()`

*page 11*    `virtual XVT_Dialog *CastToDialog()`

*page 11*    `virtual XVT_DrawableContainer*CastToDrawableContainer()`

*page 11*    `virtual XVT_Edit *CastToEdit()`

*page 11*    `virtual XVT_GroupBox *CastToGroupBox()`

*page 11*    `virtual XVT_Icon *CastToIcon()`

*page 11*    `virtual XVT_ListBox *CastToListBox()`

*page 11*    `virtual XVT_ListButton *CastToListButton()`

*page 11*    `virtual XVT_ListEdit *CastToListEdit()`

*page 11*    `virtual XVT_MenuWin *CastToMenuWin()`

*page 11*    `virtual XVT_PrintWin *CastToPrintWin()`

*page 11*    `virtual XVT_RadioButton *CastToRadioButton()`

*page 11*    `virtual XVT_ScreenWin *CastToScreenWin()`

*page 11*    `virtual XVT_ScrollBar *CastToScrollBar()`

*page 11*    `virtual XVT_StaticText *CastToStaticText()`

*page 11*    `virtual XVT_TaskWin *CastToTaskWin()`

*page 11*    `virtual XVT_TopLevelWin *CastToTopLevelWin()`

*page 12*    `virtual XVT_Rct GetInnerRect()`

*page 13*    `virtual XVT_Rct GetOuterRect()`

XVT++ 1.1

# 3

---

## XVT++ 1.1 COMPATIBILITY CLASSES

This chapter describes the XVT++ 1.1 compatibility classes and member functions.

---

## XVT++ 1.1 to 2.0 Member Function Map

This section is for programmers familiar with version 1.1 of the XVT++ Portability Toolkit; it presents the XVT++ 1.1 member functions and their corresponding XVT++ 2.0 member functions.

**Note:** Although the XVT++ 1.1 member functions are supported by XVT++ 2.0, we encourage you to upgrade your application to the latest XVT++ functionality.

The member functions are presented in the table below. The *Ret* column indicates whether the parameters (P), return value type (R), or both (PR) are identical for both functions. When both parameter and return value are identical, you should be able to simply substitute the 2.0 function name for its 1.1 counterpart. An X indicates that neither parameter nor return type is identical.

The page number for the description of the XVT++ 2.0 member function in this *Reference* is provided in the *Pg* column.

In some cases, multiple XVT++ 2.0 member functions are listed for one XVT++ 1.1 member function; for example, the function `BaseXVT::disable`. In these cases, the appropriate 2.0 member function to use depends on the type of object on which the function is operating. Refer to the member function descriptions for more information.



The following XVT++ 1.1 member functions do not have corresponding XVT++ 2.0 member functions and are not listed in the table:

BaseWin::dispatch	DlgWin::set_def	ScreenWin::set_def
BaseWin::set_timer	Font::set_font	StrList::get
BaseXVT::get_type	Rct::set	StrList::valid

<b>XVT++ 1.0 Member Function</b>	<b>XVT++ 2.0 Member Function</b>	<b>Ret</b>	<b>Pg</b>
BaseWin::get_client	XVT_Base::GetInnerRect	P	12
BaseWin::get_mask	XVT_Dialog::GetEventMask	PR	116
	XVT_DrawableContainer::GetEventMask		141
BaseWin::set_font	XVT_BaseDrawProto::SetFont	R	34
BaseWin::set_mask	XVT_Dialog::SetEventMask	PR	120
	XVT_DrawableContainer::SetEventMask		146
BaseXVT::disable	XVT_ChildBase::SetEnabledState	R	58
	XVT_Control::SetEnabledState		97
	XVT_Dialog::SetEnabledState		120
BaseXVT::enable	XVT_ChildBase::SetEnabledState	R	58
	XVT_Control::SetEnabledState		97
	XVT_Dialog::SetEnabledState		120
BaseXVT::get_rect	XVT_Base::GetOuterRect	P	13
BaseXVT::get_text	XVT_Dialog::GetTitle	X	118
	XVT_Label::GetTitle		239
	XVT_MenuWin::GetTitle		288

<b>XVT++ 1.0 Member Function</b>	<b>XVT++ 2.0 Member Function</b>	<b>Ret</b>	<b>Pg</b>
BaseXVT::hide	XVT_ChildBase::SetVisibleState	R	61
	XVT_Control::SetVisibleState		98
	XVT_Dialog::SetVisibleState		123
BaseXVT::move	XVT_Control::SetInnerRect	R	98
	XVT_Dialog::SetInnerRect		121
	XVT_DrawableContainer::SetInnerRect		148
BaseXVT::parent	XVT_ChildBase::GetParent	PR	51
	XVT_Control::GetParent		95
BaseXVT::set_text	XVT_Dialog::SetTitle	PR	122
	XVT_Label::SetTitle		240
	XVT_MenuWin::SetTitle		291
BaseXVT::show	XVT_ChildBase::SetVisibleState	PR	61
	XVT_Control::SetVisibleState		98
	XVT_Dialog::SetVisibleState		123
Control::check	XVT_CheckBox::SetCheckedState	R	45
	XVT_Toggle::SetCheckedState		320
Control::close	XVT_Control::Close	PR	92
	XVT_Dialog::Close		109
	XVT_DrawableContainer::Close		129
	XVT_TaskWin::Close		362
Control::create_def	XVT_Control::Init	R	96
	XVT_Icon::Init		240

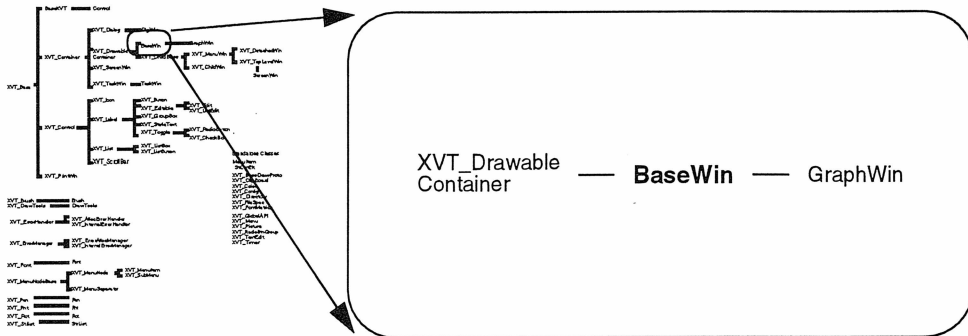
<b>XVT++ 1.0 Member Function</b>	<b>XVT++ 2.0 Member Function</b>	<b>Ret</b>	<b>Pg</b>
	XVT_Label::Init		239
	XVT_ScrollBar::Init		341
Control::create_scratch	XVT_Control::Init	R	96
	XVT_Icon::Init		230
	XVT_Label::Init		239
	XVT_ScrollBar::Init		341
Control::get_scroll_pos	XVT_ChildBase::GetScrollPosition	X	52
	XVT_ScrollBar::GetScrollPosition		339
Control::get_scroll_proportion	XVT_ChildBase::GetScrollProportion	X	52
	XVT_ScrollBar::GetScrollProportion		340
Control::get_scroll_range	XVT_ChildBase::GetScrollRange	X	53
	XVT_ScrollBar::GetScrollRange		340
Control::lbox_add	XVT_List::Add	X	244
Control::lbox_clear	XVT_List::Clear	P	245
Control::lbox_count_all	XVT_List::CountAll	P	245
Control::lbox_count_sel	XVT_List::CountSelections	P	246
Control::lbox_delete	XVT_List::Delete	X	246
Control::lbox_get_all	XVT_List::GetAll	P	247
Control::lbox_get_elt	XVT_List::GetElement	R	247
Control::lbox_get_first_sel	XVT_List::GetFirstSelection	R	248
Control::lbox_get_sel	XVT_List::GetSelections	P	249
Control::lbox_get_sel_index	XVT_List::GetSelectionIndex	P	249

<b>XVT++ 1.0 Member Function</b>	<b>XVT++ 2.0 Member Function</b>	<b>Ret</b>	<b>Pg</b>
Control::lbox_is_sel	XVT_List::GetSelectedState	R	248
Control::lbox_resume	XVT_ListBox::SetSuspendedState	R	255
Control::lbox_set_sel	XVT_List::SetSelectedState	X	250
Control::lbox_suspend	XVT_ListBox::SetSuspendedState	R	255
Control::select_text	XVT_Editable::SelectText	R	163
Control::set_scroll_pos	XVT_ChildBase::SetScrollPosition	R	59
	XVT_ScrollBar::SetScrollPosition		342
Control::set_scroll_proportion	XVT_ChildBase::SetScrollProportion	R	60
	XVT_ScrollBar::SetScrollProportion		342
Control::set_scroll_range	XVT_ChildBase::SetScrollRange	R	60
	XVT_ScrollBar::SetScrollRange		343
Control::unchecked	XVT_CheckBox::SetCheckedState	R	45
	XVT_Toggle::SetCheckedState		320
DlgWin::create	XVT_Dialog::Init	R	119
DlgWin::create_def	XVT_Dialog::Init	R	119
Font::check	XVT_MenuWin::SetFontMenu	R	289
GraphWin::arc	XVT_BaseDrawProto::DrawArc	R	17
GraphWin::get_tools	XVT_BaseDrawProto::GetBrush	P	26
	XVT_BaseDrawProto::GetPen		28
	XVT_BaseDrawProto::GetDrawTools		28
	XVT_BaseDrawProto::GetDrawMode		27
GraphWin::icon	XVT_BaseDrawProto::DrawIcon	R	18

<b>XVT++ 1.0 Member Function</b>	<b>XVT++ 2.0 Member Function</b>	<b>Ret</b>	<b>Pg</b>
GraphWin::line	XVT_BaseDrawProto::DrawALine	R	16
GraphWin::move_to	XVT_BaseDrawProto::SetCurrentPoint	R	32
GraphWin::oval	XVT_BaseDrawProto::DrawOval	R	19
GraphWin::pie	XVT_BaseDrawProto::DrawPie	R	21
GraphWin::polygon	XVT_BaseDrawProto::DrawPolygon	R	22
GraphWin::polyline	XVT_BaseDrawProto::DrawPolyline	R	23
GraphWin::rectangle	XVT_BaseDrawProto::DrawRect	R	23
GraphWin::rounded_rectangle	XVT_BaseDrawProto::DrawRoundedRect	R	24
GraphWin::set_brush	XVT_BaseDrawProto::SetBrush	R	31
GraphWin::set_font	XVT_BaseDrawProto::SetFont	R	34
GraphWin::set_mode	XVT_BaseDrawProto::SetDrawMode	PR	33
GraphWin::set_pen	XVT_BaseDrawProto::SetPen	R	36
GraphWin::set_tools	XVT_BaseDrawProto::SetDrawTools	R	34
GraphWin::text	XVT_BaseDrawProto::DrawText	R	25
MenuItem::check	XVT_MenuItem::SetCheckedState	PR	275
MenuItem::disable	XVT_MenuNode::SetEnabledState	R	279
MenuItem::enable	XVT_MenuNode::SetEnabledState	PR	279
MenuItem::uncheck	XVT_MenuItem::SetCheckedState	R	275
Rct::empty	XVT_Rct::IsEmpty	PR	329
ScreenWin::create	XVT_ChildWin::Init	R	67
	XVT_DetachedWin::Init		102
	XVT_TopLevelWin::Init		401

<b>XVT++ 1.0 Member Function</b>	<b>XVT++ 2.0 Member Function</b>	<b>Ret</b>	<b>Pg</b>
ScreenWin::create_def	XVT_ChildWin::Init	R	67
	XVT_DetachedWin::Init		102
	XVT_TopLevelWin::Init		401
ScreenWin::create_scratch	XVT_ChildWin::Init	R	67
	XVT_DetachedWin::Init		102
	XVT_TopLevelWin::Init		401
ScreenWin::get_metrics	XVT_BaseDrawProto::GetFontMetrics	X	28
StrList::add	XVT_StrList::Add	X	350
StrList::count	XVT_StrList::Count	P	352
StrList::dbg	XVT_StrList::Debug	PR	352
StrList::elt	XVT_StrList::GetElement	X	353
StrList::first	XVT_StrList::GetFirst	X	353
StrList::next	XVT_StrList::GetNext	X	354
StrList::rem	XVT_StrList::Remove	X	355
TaskWin::begin	XVT_TaskWin::Init	R	366

# BaseWin



# Overview

<b>Header File</b>	kbasewin.hpp
<b>Source File</b>	kbasewin.cc
<b>Superclass</b>	XVT_DrawableContainer
<b>Subclasses</b>	GraphWin
<b>Usage</b>	Abstract

In XVT++ 1.1, the class `BaseWin` was the base class for all windows and dialogs. It defined all the interface common to those objects.

## Member Functions

The following functions are identical to those implemented by BaseXVT:

```

page 425    virtual void disable()
page 425    virtual void enable( BOOLEAN enabled = TRUE )
page 426    WIN_DEF* get_def() const
page 426    Rct get_rect() const

```

- page 427*    virtual SSTR\* get\_text( char\* buffer, int len ) const
- page 427*    WIN\_TYPE get\_type() const
- page 428*    virtual void hide()
- page 428*    virtual void move( Rct boundary )
- page 428*    WINDOW parent()
- page 429*    void put\_def( WIN\_DEF\* In\_def )
- page 429*    virtual void set\_text( char\* str )
- page 430*    virtual void show( BOOLEAN visible = TRUE )
- The following functions are identical to those implemented by  
             XVT\_MenuWin:
- page 286*    virtual void e\_close()
- page 287*    virtual void e\_font( XVT\_Font font, FONT\_PART part )
- The following functions are identical to those implemented by  
             XVT\_ChildBase:
- page 49*    virtual void e\_hscroll( SCROLL\_CONTROL activity,  
                                 short pos )
- page 49*    virtual void e\_vscroll( SCROLL\_CONTROL activity,  
                                 short pos )
- The following function is identical to that implemented by  
             XVT\_TaskWin:
- page 364*    virtual BOOLEAN e\_quit( BOOLEAN query\_only )

---

## BaseWin::dispatch

DISPATCH AN EVENT TO THIS WINDOW

---

### Prototypes

```
long
dispatch(
    EVENT*          event )
```

### Parameters

event  
The event to be dispatched.



**Return Value**

Always 0.

**Description**

Dispatches an event to this window. The appropriate virtual event handler will be called.

**Equivalent C Function**

dispatch\_event()

---

## BaseWin::e\_activate

RECEIVE NOTIFICATION OF ACTIVATION

---

**Prototypes**

```
virtual void  
e_activate()
```

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to activation.

Calls to this function notify the object that it has gained keyboard focus and may begin receiving calls to its e\_char method.

---

## BaseWin::e\_command

RECEIVE NOTIFICATION OF A MENU SELECTION

---

**Prototypes**

```
virtual void  
e_command(  
    MenuItem          menu_item,  
    BOOLEAN           shift,  
    BOOLEAN           control )
```

**Parameters**`menu_item`

An object corresponding to the selected menu item.

`shift`

A flag that is TRUE if the shift key was held down during the menu selection, FALSE otherwise.

`control`

A flag that is TRUE if the control key was held down during the menu selection, FALSE otherwise.

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to menu selections.

Calls to this function notify the object that the user has selected an item from the associated menu.

---

**BaseWin::e\_control**

RECEIVE NOTIFICATION OF CONTROL OPERATION

---

**Prototypes**

```
virtual void
e_control(
    int          cid,
    CONTROL_INFO* info )
```

**Parameters**`cid`

The ID of the control being operated.

`info`

Information about the operation.

**Description**

This member function must be overridden by a subclass if the application wishes to take any actions in response to control operation by the user.

Calls to this function notify the object that the user has manipulated one of its contained controls.

---

## BaseWin::e\_deactivate

RECEIVE NOTIFICATION OF DEACTIVATION

---

### Prototypes

```
virtual void  
e_deactivate()
```

### Description

This member function must be overridden by a subclass if the application wishes to take any actions in response to deactivation.

Calls to this function notify the object that it has lost keyboard focus and will not receive further calls to its `e_char` method until it receives another `e_activate` call.

---

## BaseWin::get\_client

RETRIEVE THE CLIENT AREA

---

### Prototypes

```
Rct  
get_client() const
```

### Return Value

The client area rectangle. The rectangle is given in the coordinates of this window, which means that the upper left corner is always (0,0).

### Description

Retrieves the object's client area.

### Equivalent C Function

```
get_client_rect()
```

---

## BaseWin::get\_mask

RETRIEVE THE EVENT DELIVERY MASK

---

### Prototypes

```
EVENT_MASK  
get_mask() const
```

### Return Value

The current event delivery mask.

### Equivalent C Function

```
get_event_mask()
```

---

## BaseWin::get\_win

RETRIEVE THE OBJECT'S WINDOW HANDLE

---

### Prototypes

```
WINDOW  
get_win() const
```

### Return Value

This object's window handle.

---

## BaseWin::set\_font

SET THE CURRENT FONT

---

### Prototypes

```
void  
set_font(  
    FONT font,  
    BOOLEAN scale = FALSE )
```

### Parameters

```
font  
    The new current font.  
scale  
    A flag that is TRUE if the font is to be scaled, FALSE if not.
```

**Description**

Sets the current font.

**Equivalent C Function**

win\_set\_font()

---

**BaseWin::set\_mask**

SET THE EVENT DELIVERY MASK

---

**Prototypes**

```
void  
set_mask(  
    EVENT_MASK          mask )
```

**Parameters**

mask  
The new event delivery mask.

**Description**

Sets the event delivery mask.

**Equivalent C Function**

set\_event\_mask()

---

**BaseWin::set\_timer**

SET A TIMER

---

**Prototypes**

```
long  
set_timer(  
    long          interval )
```

**Parameters**

interval  
The timer interval in milliseconds.

**Return Value**

The ID of this timer.

**Description**

Sets a timer.

**Equivalent C Function**

set\_timer()

**Implementation Members**

set\_win  
set\_inited  
class\_name

**Inherited Member Functions****From XVT\_DrawableContainer**

*page 129*    void Clear()  
*page 129*    void Clear( XVT\_Color color )  
*page 129*    void Close()  
*page 128*    XVT\_BaseDrawProto\* DrawProtocol  
*page 130*    virtual void e\_char(  
                  short chr,  
                  BOOLEAN shift,  
                  BOOLEAN control)  
*page 131*    virtual void e\_create()  
*page 132*    virtual void e\_destroy()  
*page 132*    virtual void e\_focus( BOOLEAN active )  
*page 133*    virtual void e\_mouse\_dbl(  
                  XVT\_Pnt point,  
                  BOOLEAN shift,  
                  BOOLEAN control,  
                  short button )  
*page 134*    virtual void e\_mouse\_down(  
                  XVT\_Pnt point,  
                  BOOLEAN shift,  
                  BOOLEAN control,  
                  short button )

<i>page 135</i>	virtual void e_mouse_move( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_up( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 136</i>	virtual void e_size( XVT_Rct boundary )
<i>page 137</i>	virtual void e_timer( long id )
<i>page 137</i>	virtual void e_update( XVT_Rct boundary )
<i>page 139</i>	virtual long e_user( long id, void *data )
<i>page 140</i>	XVT_Control *GetCtl( long cid )
<i>page 140</i>	long GetCtlCount()
<i>page 141</i>	EVENT_MASK GetEventMask() const
<i>page 141</i>	XVT_Control *GetFirstCtl()
<i>page 142</i>	XVT_ChildBase *GetFirstWin()
<i>page 142</i>	XVT_Control *GetNextCtl()
<i>page 143</i>	XVT_ChildBase *GetNextWin()
<i>page 143</i>	long GetWinCount()
<i>page 144</i>	void Invalidate()
<i>page 144</i>	void Invalidate( XVT_Rctregion )
<i>page 145</i>	void Scroll( XVT_Rct boundary, long dh, long dv )
<i>page 146</i>	void SetEventMask( EVENT_MASK ask )
<i>page 148</i>	void SetInnerRect( XVT_Rct r )

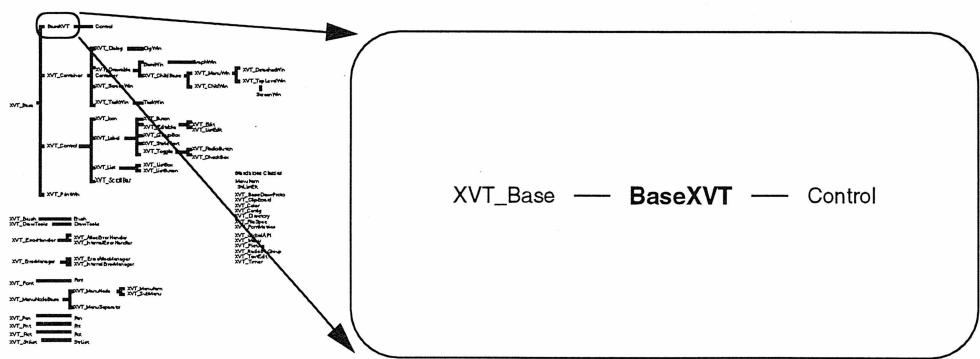
### From XVT\_Base

<i>page 11</i>	virtual BaseWin* CastToBaseWin()
<i>page 10</i>	virtual DlgWin* CastToDlgWin()
<i>page 10</i>	virtual ScreenWin* CastToScreenWin11()

<i>page 10</i>	<code>virtual TaskWin* CastToTaskWin11()</code>
<i>page 11</i>	<code>virtual XVT_Button *CastToButton()</code>
<i>page 11</i>	<code>virtual XVT_CheckBox *CastToCheckBox()</code>
<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer*CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>



# BaseXVT



## Overview

Header File	ibase.hpp
Source File	ibase.cc
Superclass	XVT_Base
Subclasses	Control
Usage	Implementation

This is the abstract class from which the XVT++ 1.1 hierarchy was derived. It provides default implementations of features common to all the various interface objects.

This class is completely compatible with the XVT++ 1.1 class of the same name.

# Member Functions

---

## BaseXVT::close

CLOSE AN OBJECT

---

### Prototypes

```
virtual void  
close()
```

### Description

Closes an object.

---

## BaseXVT::disable

DISABLE AN OBJECT

---

### Prototypes

```
virtual void  
disable()
```

### Description

Disables an object.

### Equivalent C Function

```
enable_window()
```

---

## BaseXVT::enable

ENABLE OR DISABLE AN OBJECT

---

### Prototypes

```
virtual void  
enable(  
    BOOLEAN  
    enabled = TRUE )
```

**Parameters**

enabled

A flag that is TRUE if the object is to be enabled, FALSE if it is to be disabled.

**Description**

Enables or disables an object.

**Equivalent C Function**

enable\_window()

---

**BaseXVT::get\_def**

RETRIEVE THE STORED WINDOW DEFINITION

---

**Prototypes**

WIN\_DEF\*  
get\_def() const

**Return Value**

A pointer to the stored window definition.

---

**BaseXVT::get\_rect**

RETRIEVE THE OUTER RECTANGLE

---

**Prototypes**

Rct  
get\_rect() const

**Return Value**

The object's extent rectangle relative to its parent if successful, an empty rectangle if not.

**Equivalent C Function**

get\_outer\_rect()

---

## BaseXVT::get\_text

RETRIEVE AN OBJECT'S TITLE

---

### Prototypes

```
virtual SSTR*
get_text(
    char*          buffer,
    int            len ) const
```

### Parameters

**buffer**  
A buffer to hold the object's title.

**len**  
The length of buffer in bytes.

### Return Value

buffer.

### Equivalent C Function

get\_title()

---

## BaseXVT::get\_type

RETRIEVE AN OBJECT'S WINDOW TYPE

---

### Prototypes

```
WIN_TYPE
get_type() const
```

### Return Value

The object's window type.

### Equivalent C Function

get\_window\_type()

---

## BaseXVT::hide

HIDE AN OBJECT

---

### Prototypes

```
virtual void  
hide()
```

### Description

Hides an object.

### Equivalent C Function

```
show_window()
```

---

## BaseXVT::move

MOVE AN OBJECT

---

### Prototypes

```
virtual void  
move(  
    Rct  
    boundary )
```

### Parameters

boundary  
The new size and position of the object's client area.

### Description

Moves or resizes an object.

### Equivalent C Function

```
move_window()
```

---

## BaseXVT::parent

RETRIEVE THE OBJECT'S PARENT WINDOW

---

### Prototypes

```
WINDOW  
parent()
```

**Return Value**

The window handle of the object's parent.

**Equivalent C Function**

get\_parent()

---

## BaseXVT::put\_def

SET THE STORED WINDOW DEFINITION

---

**Prototypes**

```
void
put_def(
    WIN_DEF*          In_def )
```

**Parameters**

In\_def  
The new window definition.

**Description**

Sets the stored window definition.

---

## BaseXVT::set\_text

SET AN OBJECT'S TITLE

---

**Prototypes**

```
virtual void
set_text(
    char*          str )
```

**Parameters**

str  
The new title.

**Description**

Sets an object's title.

**Equivalent C Function**

set\_title()

## BaseXVT::show

SHOW OR HIDE AN OBJECT

### Prototypes

```
virtual void
show(      BOOLEAN          visible = TRUE )
```

### Parameters

**visible**  
A flag that is TRUE if the object is to be visible, FALSE if it is to be invisible.

### Description

Shows or hides an object.

### Equivalent C Function

show\_window()

## Implementation Members

```
CloseProtocol
ShowProtocol
EnableProtocol
TitleProtocol
MoveProtocol
class_name
```

## Inherited Member Functions

### From XVT\_Base

```
page 11   virtual BaseWin* CastToBaseWin()
page 10   virtual DlgWin* CastToDlgWin()
page 10   virtual ScreenWin* CastToScreenWin11()
page 10   virtual TaskWin* CastToTaskWin11()
page 11   virtual XVT_Button *CastToButton()
page 11   virtual XVT_CheckBox *CastToCheckBox()
```

<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer *CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>





# Member Functions

## Brush::brush

CONVERT A BRUSH TO/FROM A C CBRUSH STRUCTURE

### Prototypes

```
CBRUSH  
brush()  
  
Brush  
brush(  
    CBRUSH  
    brsh )
```

### Parameters

`brsh`  
The C CBRUSH structure to convert from.

### Return Value

`brush()`  
The equivalent C CBRUSH structure.  
`brush( brsh )`  
This brush.

### Description

`brush()`  
Converts a brush to a C CBRUSH structure.  
`brush( brsh )`  
Converts a C CBRUSH structure to a brush.

### Implementation Notes

The `brush( brsh )` form of the call is a little odd in that it modifies the brush *and* returns a copy of the modified brush.

---

## Brush::color

RETRIEVE OR SET THE BRUSH'S COLOR

---

### Prototypes

```
COLOR  
color()  
  
void  
color(    COLOR                clr )
```

### Parameters

clr  
The brush's new color.

### Return Value

The brush's current color.

### Description

```
color()  
    Retrieves the current color.  
color( clr )  
    Sets the brush's current color.
```

---

## Brush::pat

RETRIEVE OR SET THE BRUSHES' PATTERN

---

### Prototypes

```
PAT_STYLE  
pat()  
  
void  
pat(    PAT_STYLE                p )
```

### Parameters

p  
The brush's new pattern.

### Return Value

The brush's current pattern.

**Description**

pat()  
Retrieves the brush's current pattern.

pat( p )  
Sets the brush's current pattern.

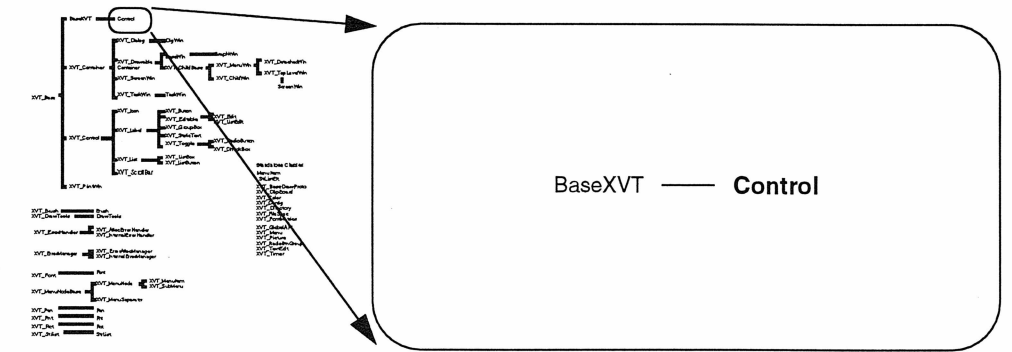
**Implementation Members**

char\* class\_name()

**Inherited Member Functions****From XVT\_Brush**

*page 38*    XVT\_Color GetColor()  
*page 38*    PAT\_STYLE GetPattern()  
*page 39*    void SetColor( XVT\_Color c )  
*page 39*    void SetPattern( PAT\_STYLE p )

# Control



## Overview

Header File	kctl.hpp
Source File	kctl.cc
Superclass	BaseXVT
Subclasses	
Usage	Concrete

Instances of the Control class represent controls.  
This class is fully compatible with the XVT++ 1.1 Control class.

## Constructors

```
Control( XVT_Base* parent = NULL, int cid = 0,  
        CONTROL_INFO* ci = NULL )  
virtual ~Control()
```

# Member Functions

---

## Control::check

CHECK A CHECK BOX OR RADIO BUTTON

---

### Prototypes

```
void  
check(  
    BOOLEAN                check_me = TRUE )  
  
void  
check(  
    int                    i1,  
    int                    i2 )  
  
void  
check(  
    Control*               ctlarray[],  
    int                    numctls )
```

### Parameters

**check\_me**  
A flag that is TRUE if the button is to be checked, FALSE if not.

**i1**  
The control ID of the first radio button in the group.

**i2**  
The control ID of the last radio button in the group.

**ctlarray**  
The array of radio buttons in the group.

**numctls**  
The number of controls in ctlarray.

### Description

**check( check\_me = TRUE )**  
Checks a check box.

**check( i1, i2 )**  
Checks this radio button. The group of which this button is a part is given bounded by the two IDs given.

**check( ctlarray[], numctls )**  
Checks this radio button. The group of which this button is a part is defined by the given array of controls.

**Equivalent C Function**

```
win_check_box()
win_check_radio_button()
```

---

**Control::close**

DESTROY A CONTROL

---

**Prototypes**

```
void
close()
```

**Description**

Destroys a control in a window.

**Equivalent C Function**

```
close_window()
```

---

**Control::create\_def**

CREATE A CONTROL FROM A WIN\_DEF STRUCTURE

---

**Prototypes**

```
virtual BOOLEAN
create_def(
    XVT_Base*          parent_win,
    long               appdata = 0L)
```

**Parameters**

```
parent_win
    The new control's parent window.
appdata
    The new control's application data.
```

**Return Value**

TRUE if the control was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by the control. Recovery can be attempted by disposing of the new control, closing another control, and retrying the creation of the control.

## Description

Creates a control from a WIN\_DEF structure. The WIN\_DEF structure is stored with the object, and must be a result of a prior put\_def. The created control can be put only into a window, not a dialog.

## Equivalent C Function

```
create_def_control().
```

---

# Control::create\_scratch

CREATE A CONTROL FROM PARAMETERS

---

## Prototypes

```
virtual BOOLEAN
create_scratch(
    XVT_Base*      parent_win,
    WIN_TYPE       wtype,
    int            ctrl_id,
    Rct            rct,
    long           ctl_flags,
    SSTR*          title,
    long           appdata = 0L)
```

## Parameters

**parent\_win**  
The new control's parent window.

**wtype**  
The new control's window type.

**ctrl\_id**  
The new control's ID.

**rct**  
The new control's boundary rectangle.

**ctl\_flags**  
The new control's attribute flags.

**title**  
The new control's title.

**appdata**  
The new control's application data.

## Return Value

TRUE if the control was successfully created, FALSE otherwise. A FALSE return value means that the native system ran out of some resource that is consumed by the control. Recovery can be attempted



by disposing of the new control, closing another control, and retrying the creation of the control.

### Description

Creates a control strictly from input parameters. The control may be put only in a window, not a dialog.

### Equivalent C Function

create\_control()

---

## Control::get\_scroll\_pos

RETRIEVE A SCROLLBAR'S THUMB POSITION.

---

### Prototypes

```
int  
get_scroll_pos() const
```

### Return Value

The scrollbar's current thumb position.

### Equivalent C Function

get\_scroll\_pos()

---

## Control::get\_scroll\_proportion

RETRIEVE A SCROLLBAR THUMB'S PROPORTION

---

### Prototypes

```
int  
get_scroll_proportion() const
```

### Return Value

The scrollbar thumb's proportion.

### Equivalent C Function

get\_scroll\_proportion()

---

## Control::get\_scroll\_range

RETRIEVE A SCROLLBAR'S RANGE

---

### Prototypes

```
void  
get_scroll_range(  
    int* min,  
    int* max ) const
```

### Parameters

**min**  
A pointer to storage for the minimum of the range.

**max**  
A pointer to storage for the maximum of the range.

### Equivalent C Function

```
get_scroll_range()
```

---

## Control::id

RETRIEVE A CONTROL'S ID

---

### Prototypes

```
int  
id() const
```

### Return Value

The control's ID.

---

## Control::is\_checked

DETERMINE IF A CHECK BOX OR RADIO BUTTON IS CHECKED

---

### Prototypes

```
BOOLEAN  
is_checked() const
```

**Return Value**

A flag that is TRUE if the button is checked, FALSE if not.

---

**Control::lbox\_add**

ADD AN ITEM OR ITEMS TO A LIST BOX

---

**Prototypes**

```

BOOLEAN
lbox_add(
    int          index,
    SSTR*        s )

BOOLEAN
lbox_add(
    SSTR*        s )

BOOLEAN
lbox_add(
    int          index,
    SLIST        sl )

BOOLEAN
lbox_add(
    SLIST        sl )

```

**Parameters**

**index**  
The index of the element before which the new element is to be added.

**s**  
The string defining the new element.

**sl**  
The string list defining the new elements.

**Return Value**

A flag that is TRUE if the operation was successful, FALSE if not.

**Description**

```

lbox_add( index, s )
    Adds a string to a list box.

lbox_add( s )
    Adds a string to the end of a list box.

```

```
lbox_add( index, sl )  
    Adds a list of strings to a list box.  
lbox_add( sl )  
    Adds a list of strings to the end of a list box.
```

**Equivalent C Function**

```
win_list_add()
```

---

**Control::lbox\_clear**

REMOVE ALL ITEMS FROM A LIST BOX

---

**Prototypes**

```
BOOLEAN  
lbox_clear()
```

**Return Value**

A flag that is TRUE if the operation was successful, FALSE if not.

**Description**

Removes all items from a list box.

**Equivalent C Function**

```
win_list_clear()
```

---

**Control::lbox\_count\_all**

RETRIEVE THE NUMBER OF ITEMS IN A LIST BOX

---

**Prototypes**

```
int  
lbox_count_all() const
```

**Return Value**

The number of items in a list box.

**Equivalent C Function**

```
win_list_count_all()
```

---

## Control::lbox\_count\_sel

RETRIEVE THE NUMBER OF SELECTED ITEMS

---

### Prototypes

```
int  
lbox_count_sel() const
```

### Return Value

The number of selected items.

### Equivalent C Function

```
win_list_count_sel()
```

---

## Control::lbox\_delete

REMOVE AN ITEM

---

### Prototypes

```
BOOLEAN  
lbox_delete(  
    int  
            index )
```

### Parameters

```
index  
    The index of the item to delete.
```

### Return Value

A flag that is TRUE if the operation was successful, FALSE if not.

### Equivalent C Function

```
win_list_delete()
```

---

## Control::lbox\_get\_all

RETRIEVE ALL ITEMS

---

### Prototypes

```
SLIST  
lbox_get_all() const
```

**Return Value**

An SLIST containing all of the items in the list box.

**Equivalent C Function**

win\_list\_get\_all()

---

**Control::lbox\_get\_elt**

RETRIEVE A LIST BOX ELEMENT

---

**Prototypes**

```
BOOLEAN  
lbox_get_elt(  
    int                index,  
    SSTR*              s,  
    int                sz_s )
```

**Parameters**

index  
The element index. Zero is first.

s  
A buffer to receive the item text.

sz\_s  
The size of the buffer pointed to by s.

**Return Value**

A flag that is TRUE if the operation was successful, FALSE if not.

**Equivalent C Function**

win\_list\_get\_elt()

---

**Control::lbox\_get\_first\_sel**

RETRIEVE THE FIRST SELECTED ITEM

---

**Prototypes**

```
BOOLEAN  
lbox_get_first_sel(  
    SSTR*              s,  
    int                sz_s )
```

**Parameters**

`s`  
A buffer to receive the item text.

`sz_s`  
The size of the buffer pointed to by `s`.

**Return Value**

A flag that is TRUE if the operation was successful, FALSE if not.

**Equivalent C Function**

`win_list_get_first_sel()`

---

## Control::lbox\_get\_sel

RETRIEVE ALL SELECTED ITEMS

---

**Prototypes**

`SLIST`  
`lbox_get_sel() const`

**Return Value**

An `SLIST` containing all selected items.

**Equivalent C Function**

`win_list_get_sel()`

---

## Control::lbox\_get\_sel\_index

RETRIEVE THE INDEX OF THE FIRST SELECTED ITEM

---

**Prototypes**

`int`  
`lbox_get_sel_index() const`

**Return Value**

The index of the first selected item.

**Equivalent C Function**

`win_list_get_sel_index()`

---

## Control::lbox\_is\_sel

DETERMINE IF AN ITEM IS SELECTED

---

### Prototypes

```
BOOLEAN  
lbox_is_sel(  
    int                index ) const
```

### Parameters

index  
The index of the item to check for selectedness.

### Return Value

A flag that is TRUE if the given item was selected, FALSE if not.

### Equivalent C Function

```
win_list_is_sel()
```

---

## Control::lbox\_resume

RESUME UPDATES TO A LIST BOX

---

### Prototypes

```
void  
lbox_resume()
```

### Description

Resumes updates to a list box. Cancels a previous call to lbox\_suspend.

### Equivalent C Function

```
win_list_resume()
```



---

## Control::lbox\_set\_sel

SELECT AN ITEM

---

### Prototypes

```

    BOOLEAN
    lbox_set_sel(
        int          index,
        BOOLEAN      select )

    BOOLEAN
    lbox_set_sel(
        BOOLEAN      select )

```

### Parameters

**index**  
The index of the item to select.

**select**  
A flag that is TRUE if the item is to be selected, FALSE if unselected.

### Return Value

A flag that is TRUE if the operation was successful, FALSE if not.

### Description

```

lbox ( index, select )
    Selects/unselects the given item.

lbox ( select )
    Selects/unselects all items.

```

### Equivalent C Function

```
win_list_set_sel()
```

---

## Control::lbox\_suspend

SUSPEND UPDATES TO A LIST BOX

---

### Prototypes

```

void
lbox_suspend()

```

### Description

Suspends updates to a list box.

**Equivalent C Function**

```
win_list_suspend()
```

---

**Control::select\_text**

SELECT TEXT

---

**Prototypes**

```
void
select_text(
    int          from,
    int          to )
```

**Parameters**

**from**  
The first character index of the desired selection.

**to**  
The last character index of the desired selection.

**Description**

Selects text in an edit control.

**Equivalent C Function**

```
win_select_item_text()
```

---

**Control::set\_scroll\_pos**

SET A SCROLLBAR'S THUMB POSITION

---

**Prototypes**

```
void
set_scroll_pos(
    int          pos )
```

**Parameters**

**pos**  
The new thumb position.

**Description**

Sets a scrollbar's thumb position.

**Equivalent C Function**

```
set_scroll_pos()
```

---

**Control::set\_scroll\_proportion**

SET A SCROLLBAR'S THUMB PROPORTION

---

**Prototypes**

```
void  
set_scroll_proportion(  
    int                proportion )
```

**Parameters**

proportion  
The new thumb proportion.

**Description**

Sets a scrollbar's thumb proportion.

**Equivalent C Function**

```
set_scroll_proportion()
```

---

**Control::set\_scroll\_range**

SET A SCROLLBAR'S RANGE

---

**Prototypes**

```
void  
set_scroll_range(  
    int                min,  
    int                max )
```

**Parameters**

min  
The minimum of the new range.

max  
The maximum of the new range.

**Description**

Sets a scrollbar's range.

**Equivalent C Function**

```
set_scroll_range()
```

---

**Control::unchecked**

UNCHECK A CHECK BOX

---

**Prototypes**

```
void  
unchecked()
```

**Description**

Unchecks a check box. Equivalent to `check( FALSE )`.

**Equivalent C Function**

```
win_check_box()
```

**Implementation Members**

```
class_name  
in_dialog  
RealControl  
GetRealControl
```

**Inherited Member Functions****From BaseXVT**

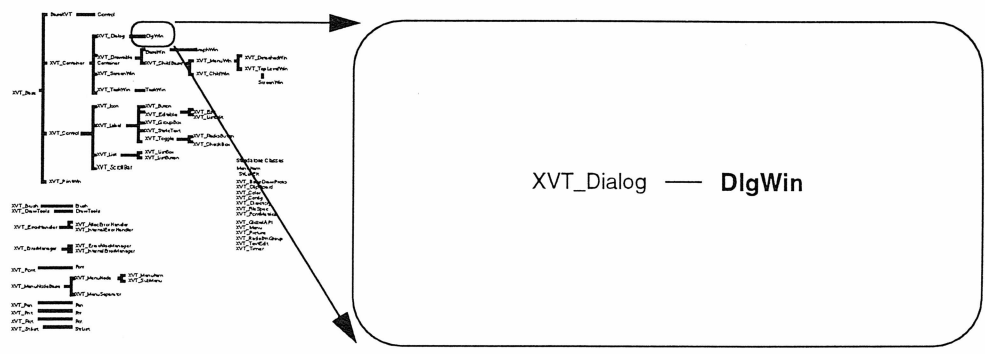
*page 425*    virtual void disable()  
*page 425*    virtual void enable( BOOLEAN v = TRUE )  
*page 426*    WIN\_DEF\* get\_def()  
*page 426*    Rct get\_rect() const  
*page 427*    virtual char\* get\_text( char \*, int )  
*page 427*    WIN\_TYPE get\_type() const  
*page 428*    virtual void hide()  
*page 428*    virtual void move( Rct r )  
*page 428*    WINDOW parent()

*page 429*    `void put_def( WIN_DEF* In_def )`  
*page 429*    `virtual void set_text( char * ch)`  
*page 430*    `virtual void show( BOOLEAN v = TRUE )`

### **From XVT\_Base**

*page 11*    `virtual BaseWin* CastToBaseWin()`  
*page 10*    `virtual DlgWin* CastToDlgWin()`  
*page 10*    `virtual ScreenWin* CastToScreenWin11()`  
*page 10*    `virtual TaskWin* CastToTaskWin11()`  
*page 11*    `virtual XVT_Button *CastToButton()`  
*page 11*    `virtual XVT_CheckBox *CastToCheckBox()`  
*page 11*    `virtual XVT_ChildWin *CastToChildWin()`  
*page 11*    `virtual XVT_DetachedWin *CastToDetachedWin()`  
*page 11*    `virtual XVT_Dialog *CastToDialog()`  
*page 11*    `virtual XVT_DrawableContainer*CastToDrawableContainer()`  
*page 11*    `virtual XVT_Edit *CastToEdit()`  
*page 11*    `virtual XVT_GroupBox *CastToGroupBox()`  
*page 11*    `virtual XVT_Icon *CastToIcon()`  
*page 11*    `virtual XVT_ListBox *CastToListBox()`  
*page 11*    `virtual XVT_ListButton *CastToListButton()`  
*page 11*    `virtual XVT_ListEdit *CastToListEdit()`  
*page 11*    `virtual XVT_MenuWin *CastToMenuWin()`  
*page 11*    `virtual XVT_PrintWin *CastToPrintWin()`  
*page 11*    `virtual XVT_RadioButton *CastToRadioButton()`  
*page 11*    `virtual XVT_ScreenWin *CastToScreenWin()`  
*page 11*    `virtual XVT_ScrollBar *CastToScrollBar()`  
*page 11*    `virtual XVT_StaticText *CastToStaticText()`  
*page 11*    `virtual XVT_TaskWin *CastToTaskWin()`  
*page 11*    `virtual XVT_TopLevelWin *CastToTopLevelWin()`  
*page 12*    `virtual XVT_Rct GetInnerRect()`  
*page 13*    `virtual XVT_Rct GetOuterRect()`

# DlgWin



## Overview

Header File	<code>kdlg.hpp</code>
Source File	<code>kdlg.cc</code>
Superclass	<code>XVT_Dialog</code>
Subclasses	
Usage	Abstract

The `DlgWin` class defines the interface to all dialogs.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use `XVT_Dialog` instead.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

```
DlgWin()
virtual ~DlgWin()
```

## Member Functions

The following functions are identical to those implemented by BaseXVT:

<i>page 425</i>	<code>virtual void disable()</code>
<i>page 425</i>	<code>virtual void enable( BOOLEAN v = TRUE )</code>
<i>page 426</i>	<code>WIN_DEF* get_def() const</code>
<i>page 426</i>	<code>Rct get_rect() const</code>
<i>page 427</i>	<code>virtual SSTR* get_text( SSTR*, int ) const</code>
<i>page 427</i>	<code>WIN_TYPE get_type() const</code>
<i>page 428</i>	<code>virtual void hide()</code>
<i>page 428</i>	<code>virtual void move( Rct r )</code>
<i>page 428</i>	<code>WINDOW parent()</code>
<i>page 429</i>	<code>void put_def( WIN_DEF* In_def )</code>
<i>page 429</i>	<code>virtual void set_text( SSTR* ch)</code>
<i>page 430</i>	<code>virtual void show( BOOLEAN v = TRUE )</code>

---

## DlgWin::create

CREATE A DIALOG FROM RESOURCES

---

### Prototypes

```
virtual BOOLEAN
create(
    int          rid,
    WIN_TYPE     wtype = WD_MODAL,
    long         data = 0L )
```

### Parameters

**rid**  
The dialog resource ID.

**wtype**  
The type of dialog, WD\_MODAL or WD\_MODELESS.

**userdata**  
The user data associated with this dialog.

**Return Value**

A flag that is TRUE if the operation succeeded, FALSE if it failed.

**Description**

Creates a dialog from resources.

**Equivalent C Function**

create\_res\_dialog()

---

## DlgWin::create\_def

CREATE A DIALOG FROM A DEFINITION

---

**Prototypes**

```
virtual BOOLEAN  
create_def(  
    long  
    userdata )
```

**Parameters**

userdata  
The user data associated with this dialog.

**Return Value**

A flag that is TRUE if the operation succeeded, FALSE if it failed.

**Description**

Creates a dialog from a definition.

**Equivalent C Function**

create\_def\_dialog()

---

## DlgWin::set\_def

SET THE ASSOCIATED WIN\_DEF FROM A RESOURCE

---

**Prototypes**

```
virtual BOOLEAN  
set_def(  
    int  
    rid )
```



**Parameters**

`rid`  
The resource ID from which to get the dialog definition.

**Return Value**

A flag that is TRUE if the operation succeeded, FALSE if it failed.

**Description**

Sets the associated WIN\_DEF from a resource.

**Equivalent C Function**

`get_res_dialog()`

## Implementation Members

`class_name`

## Inherited Member Functions

**From XVT\_Dialog**

*page 109*    `void Close()`

*page 109*    `virtual void e_char(`  
                  `short chr,`  
                  `BOOLEAN shift,`  
                  `BOOLEAN control)`

*page 111*    `virtual void e_close()`

*page 111*    `virtual void e_create()`

*page 112*    `virtual void e_destroy()`

*page 112*    `virtual void e_focus( BOOLEAN active )`

*page 113*    `virtual void e_size( short width, short height )`

*page 114*    `virtual void e_timer( long id )`

*page 115*    `virtual long e_user( long id, void *data )`

*page 115*    `XVT_Control *GetCtl( long cid )`

*page 116*    `long GetCtlCount()`

*page 116*    `BOOLEAN GetEnabledState()`

*page 116*    `EVENT_MASK GetEventMask() const`

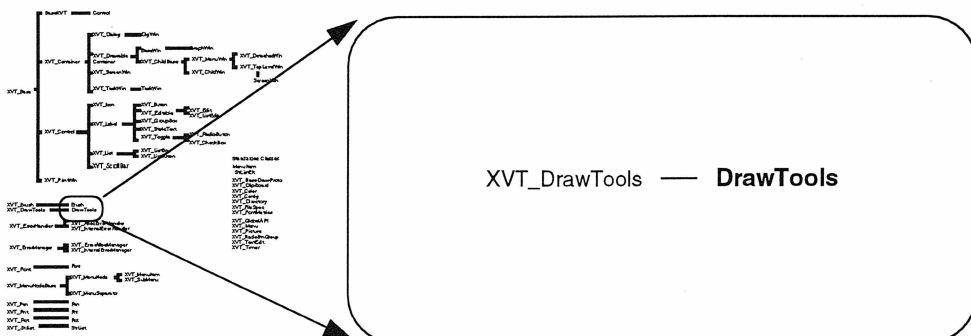
*page 117*    XVT\_Control \*GetFirstCtl()  
*page 117*    XVT\_Control \*GetNextCtl()  
*page 118*    void GetTitle( char \*buffer, long len )  
*page 118*    BOOLEAN GetVisibleState()  
*page 119*    BOOLEAN Init( long rid )  
*page 119*    BOOLEAN Init( XVT\_DialogDef \*def )  
*page 120*    void SetEnabledState( BOOLEAN state )  
*page 120*    void SetEventMask( EVENT\_MASK ask )  
*page 121*    void SetInnerRect( XVT\_Rct rect )  
*page 122*    void SetTitle( char \*str )  
*page 123*    void SetVisibleState( BOOLEAN )

### From XVT\_Base

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()

*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# DrawTools



## Overview

<b>Header File</b>	ktool.hpp
<b>Source File</b>	ktool.cc
<b>Superclass</b>	XVT_DrawTools
<b>Subclasses</b>	
<b>Usage</b>	Concrete

Instances of this class define how drawing primitives are rendered in a window. Each window maintains an instance of this class.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use XVT\_DrawTools instead.

## Constructors

```

DrawTools()
DrawTools(Pen pn )
DrawTools(Pen pn, Brush brsh )
DrawTools(Pen pn, Brush brsh, Font fnt,
          DRAW_MODE mde = M_COPY)

```

## Member Functions

---

### DrawTools::brush

GET OR SET THE BRUSH

---

#### Prototypes

```
Brush  
brush()  
  
void  
brush(  
    Brush  
    brsh )
```

#### Parameters

brsh  
The new brush.

#### Return Value

brush()  
The draw tools' brush.

#### Description

brush( brsh )  
Sets the draw tools' brush.

---

### DrawTools::font

GET OR SET THE FONT

---

#### Prototypes

```
Font  
font()  
  
void  
font(  
    Font  
    fnt )
```

#### Parameters

fnt  
The new font.

**Return Value**

font()  
The draw tools' font.

**Description**

font( fnt )  
Sets the draw tools' font.

---

**DrawTools::mode**

SET OR GET THE DRAW TOOLS' DRAWING MODE

---

**Prototypes**

```

DRAW_MODE
mode()

void
mode( DRAW_MODE mde )

```

**Parameters**

mde  
The new drawing mode.

**Return Value**

mode()  
The draw tools' drawing mode.

**Description**

mode(mde )  
Sets the draw tools' drawing mode.

---

**DrawTools::pen**

GET OR SET THE PEN

---

**Prototypes**

```

Pen
pen()

void
pen( Pen pen )

```

**Parameters**

pen  
The new pen.

**Return Value**

pen()  
The draw tools' pen.

**Description**

pen( pen)  
Sets the draw tools' pen.

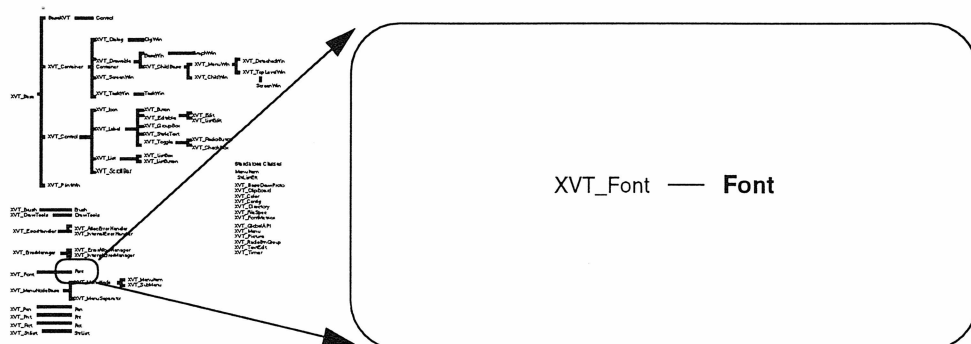
**Implementation Members**

class\_name  
ControlEvent

**Inherited Member Functions****From XVT\_DrawTools**

page 151 XVT\_Color GetBackColor()  
 page 151 XVT\_Brush GetBrush()  
 page 152 XVT\_Font GetFont()  
 page 152 XVT\_Color GetForeColor()  
 page 152 DRAW\_MODE GetMode()  
 page 153 BOOLEAN GetOpaqueText()  
 page 153 XVT\_Pen GetPen()  
 page 153 void SetBackColor( XVT\_Color c )  
 page 154 void SetBrush( XVT\_Brush b )  
 page 154 void SetFont( XVT\_Font f )  
 page 155 void SetForeColor( XVT\_Color c )  
 page 155 void SetMode( DRAW\_MODE mode )  
 page 156 void SetOpaqueText( BOOLEAN ot )  
 page 157 void SetPen( XVT\_Pen p )

# Font



## Overview

Header File	kttool.hpp
Source File	kttool.cc
Superclass	XVT_Font
Subclasses	
Usage	Concrete

Instances of this class define how text drawing primitives are rendered in a window. Each window maintains an instance of this class.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use XVT\_Font instead.

## Constructors

```
Font()
Font( FONT* font_ptr )
```



## Member Functions

---

### Font::check

CHECK THE MENU ITEM CORRESPONDING TO THIS FONT

---

#### Prototypes

```
void  
check(  
    WINDOW  
    win )
```

#### Parameters

win  
The window whose font menu is checked.

#### Description

Checks the menu item corresponding to this font.

#### Equivalent C Function

```
win_set_font_menu()
```

---

### Font::family

GET OR SET A FONT'S FAMILY

---

#### Prototypes

```
FONT_FAMILY*  
family()  
  
void  
family(  
    int  
    family )
```

#### Parameters

family  
The font's new family.

#### Return Value

```
family()  
The font's family.
```

**Description**

```
family( family )  
    Sets the font's family.
```

---

**Font::get\_font**

RETRIEVE THE XVT LIBRARY FONT STRUCTURE

---

**Prototypes**

```
FONT*  
get_font()
```

**Return Value**

The C FONT structure corresponding to this Font.

---

**Font::set\_font**

SELECT A FONT BASED ON FAMILY, SIZE, AND STYLE

---

**Prototypes**

```
void  
set_font(  
    short          sz,  
    int            fam,  
    int            st )
```

**Parameters**

**sz**  
The size of the desired font.

**fam**  
The family of the desired font.

**st**  
The style of the desired font.

**Description**

Selects a font based on family, size and style.

**Equivalent C Function**

```
select_font()
```

---

## Font::size

GET OR SET A FONT'S SIZE

---

### Prototypes

```
short  
size()  
  
void  
size(  
    short                sz )
```

### Parameters

sz  
The font's new size.

### Return Value

size()  
The font's size.

### Description

```
size( short sz )  
Sets the font's size.
```

---

## Font::style

GET OR SET A FONT'S STYLE

---

### Prototypes

```
FONT_STYLE*  
style()  
  
void  
style(  
    int                style )
```

### Parameters

style  
The font's new style.

### Return Value

style()  
The font's style.

**Description**

```
style( style )  
Sets the font's style.
```

**Implementation Members**

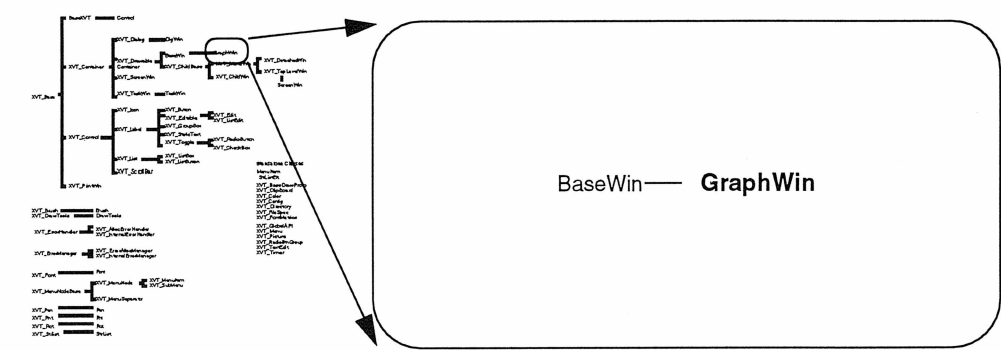
```
class_name
```

**Inherited Member Functions****From XVT\_Font**

```
page 176    short GetSize() const
```

```
page 176    void SetSize( short size )
```

# GraphWin



## Overview

Header File	kgraph.hpp
Source File	kgraph.cc
Superclass	BaseWin
Subclasses	
Usage	Abstract

The GraphWin class defines the drawing interface for windows.

This class is completely compatible with the XVT++ 1.1 class of the same name.

## Member Functions

The following functions are identical to those implemented by BaseXVT:

- page 425 virtual void disable()
- page 425 virtual void enable( BOOLEAN enabled = TRUE )
- page 426 WIN\_DEF\* get\_def()

*page 426*    `Rct get_rect() const`  
*page 427*    `virtual char* get_text( char* buffer, int len )`  
*page 427*    `WIN_TYPE get_type() const`  
*page 428*    `virtual void hide()`  
*page 428*    `virtual void move( Rct boundary )`  
*page 428*    `WINDOW parent()`  
*page 429*    `void put_def( WIN_DEF* in_def )`  
*page 429*    `virtual void set_text( char* str )`  
*page 430*    `virtual void show( BOOLEAN visible = TRUE )`

---

## GraphWin::arc

DRAW AN ARC

---

### Prototypes

```

void
arc(
    Rct
    Pnt
    Pnt
    lrct,
    start,
    stop )

```

### Parameters

```

lrct
    The bounding rectangle.
start
    The start point.
stop
    The stop point.

```

### Description

Draws an arc.

### Equivalent C Function

`win_draw_arc()`

---

## GraphWin::clear

CLEAR THE WINDOW

---

### Prototypes

```
void  
clear()
```

### Description

Clears the window.

### Implementation Notes

Note that this function is different from the `Clear` function defined elsewhere in XVT++. This version always clears the window with white.

---

## GraphWin::get\_tools

RETRIEVE THE CURRENT DRAWING TOOLS

---

### Prototypes

```
DrawTools  
get_tools() const
```

### Return Value

A copy of the current draw tools.

### Equivalent C Function

```
win_get_draw_ctools()
```

---

## GraphWin::icon

DRAW AN ICON

---

### Prototypes

```
void  
icon(  
    Pnt  
    int  
    p,  
    rid )
```

**Parameters**

`p`  
The location of the icon's upper-left corner.

`rid`  
The icon's resource ID.

**Description**

Draws an icon.

**Equivalent C Function**

`win_draw_icon()`

---

## GraphWin::line

DRAW A LINE

---

**Prototypes**

```
void
line(
    Pnt      from,
    Pnt      to,
    BOOLEAN  start_arrow = FALSE,
    BOOLEAN  end_arrow = FALSE)
```

**Parameters**

`from`  
The starting point of the line.

`to`  
The end point of the line.

`start_arrow`  
A flag that is TRUE if the line is to start with an arrow, FALSE if not.

`end_arrow`  
A flag that is TRUE if the line is to end with an arrow, FALSE if not.

**Description**

Draws a line with or without arrows.

**Equivalent C Function**

`win_draw_aline()`



---

## GraphWin::move\_to

MOVE THE CURRENT POSITION

---

### Prototypes

```
void  
move_to(  
    Pnt                p )
```

### Parameters

p  
The new current position.

### Description

Moves the current position.

### Equivalent C Function

```
win_move_to()
```

---

## GraphWin::oval

DRAW AN OVAL

---

### Prototypes

```
void  
oval(  
    Rct                r )
```

### Parameters

r  
The bounding rectangle.

### Description

Draws an oval.

### Equivalent C Function

```
win_draw_oval()
```

---

## GraphWin::pie

DRAW A PIE

---

### Prototypes

```
void  
pie(  
    Rct  
    Pnt  
    Pnt  
    r,  
    start,  
    stop )
```

### Parameters

r  
The bounding rectangle.

start  
The start point.

stop  
The stop point.

### Description

Draws a pie.

### Equivalent C Function

win\_draw\_pie()

---

## GraphWin::polygon

DRAW A POLYGON

---

### Prototypes

```
void  
polygon(  
    Pnt*  
    int  
    points,  
    npoints )
```

### Parameters

points  
An array of points.

npoints  
The number of points in points.

**Description**

Draws a polygon.

**Equivalent C Function**

win\_draw\_polygon()

---

## GraphWin::polyline

DRAW A POLYLINE

---

**Prototypes**

```
void  
polyline(  
    Pnt*      points,  
    int       npoints )
```

**Parameters**

points  
    An array of points.

npoints  
    The number of points in points.

**Description**

Draws a polyline.

**Equivalent C Function**

win\_draw\_polyline()

---

## GraphWin::rectangle

DRAW A RECTANGLE

---

**Prototypes**

```
void  
rectangle(  
    Rct      r )
```

**Parameters**

r  
    The rectangle.

**Description**

Draws a rectangle.

**Equivalent C Function**

win\_draw\_rect()

---

## GraphWin::rounded\_rectangle

DRAW A RECTANGLE WITH ROUNDED CORNERS

---

**Prototypes**

```
void  
rounded_rectangle(  
    Rct  
    int  
    int  
    r,  
    oval_width = 20,  
    oval_height = 20 )
```

**Parameters**

r  
The rectangle.

oval\_width  
The width of the corner oval.

oval\_height  
The height of the corner oval.

**Description**

Draws a rectangle with rounded corners.

**Equivalent C Function**

win\_draw\_roundrect()

---

## GraphWin::set\_brush

SET THE CURRENT BRUSH

---

**Prototypes**

```
void  
set_brush(  
    Brush  
    b )
```

**Parameters**

b  
The new brush.

**Description**

Sets the current brush.

**Equivalent C Function**

win\_set\_cbrush()

---

## GraphWin::set\_font

SET THE CURRENT FONT

---

**Prototypes**

```
void  
set_font(  
    Font font )
```

**Parameters**

font  
The new font.

**Description**

Sets the current font.

**Equivalent C Function**

win\_set\_font()

---

## GraphWin::set\_mode

SET THE CURRENT DRAWING MODE

---

**Prototypes**

```
void  
set_mode(  
    DRAW_MODE mode )
```

**Parameters**

mode  
The new drawing mode.

**Description**

Sets the current drawing mode.

**Equivalent C Function**

win\_set\_draw\_mode()

---

**GraphWin::set\_pen**

SET THE CURRENT PEN

---

**Prototypes**

```
void  
set_pen(  
    Pen  
    p )
```

**Parameters**

p  
The new pen.

**Description**

Sets the current pen.

**Equivalent C Function**

win\_set\_cpen()

---

**GraphWin::set\_tools**

SET THE CURRENT DRAWING TOOLS

---

**Prototypes**

```
void  
set_tools(  
    DrawTools  
    tools )
```

**Parameters**

tools  
The new drawing tools.

**Description**

Sets the current drawing tools.

### Equivalent C Function

```
win_set_draw_ctools()
```

## GraphWin::text

## DRAW A TEXT STRING

## Prototypes

```
void
text(
    Pnt
    char*
    int
    p,
    str,
    len )
```

## Parameters

p	The location of the start of the string's baseline.
str	The string.
len	The length of the string in bytes, or $-1$ if the string is null-terminated and the entire string is to be drawn.

### Description

Draws a text string.

### Equivalent C Function

win\_draw\_text()

## Implementation Members

class\_name

## Inherited Member Functions

## From BaseWin

```

page 415    long dispatch( EVENT * )
page 416    virtual void e_activate()
page 416    virtual void e_command( MenuItem i, BOOLEAN shift,
    BOOLEAN control )

```

*page 417*    virtual void e\_control( int cid, CONTROL\_INFO\* info )  
*page 418*    virtual void e\_deactivate()  
*page 418*    Rct get\_client()  
*page 419*    EVENT\_MASK get\_mask()  
*page 419*    WINDOW get\_win()  
*page 419*    void set\_font( Font, BOOLEAN )  
*page 420*    void set\_mask( EVENT\_MASK mask )  
*page 420*    long set\_timer( long interval )

### **From XVT\_DrawableContainer**

*page 129*    void Clear()  
*page 129*    void Clear( XVT\_Color color )  
*page 129*    void Close()  
*page 128*    XVT\_BaseDrawProto\* DrawProtocol  
*page 130*    virtual void e\_char(  
             short chr,  
             BOOLEAN shift,  
             BOOLEAN control)  
*page 131*    virtual void e\_create()  
*page 132*    virtual void e\_destroy()  
*page 132*    virtual void e\_focus( BOOLEAN active )  
*page 133*    virtual void e\_mouse\_dbl(  
             XVT\_Pnt point,  
             BOOLEAN shift,  
             BOOLEAN control,  
             short button )  
*page 134*    virtual void e\_mouse\_down(  
             XVT\_Pnt point,  
             BOOLEAN shift,  
             BOOLEAN control,  
             short button )  
*page 135*    virtual void e\_mouse\_move(  
             XVT\_Pnt point,  
             BOOLEAN shift,  
             BOOLEAN control,  
             short button )



*page 135*    virtual void e\_mouse\_up(  
                   XVT\_Pnt point,  
                   BOOLEAN shift,  
                   BOOLEAN control,  
                   short button )  
*page 136*    virtual void e\_size( XVT\_Rct boundary )  
*page 137*    virtual void e\_timer( long id )  
*page 137*    virtual void e\_update( XVT\_Rct boundary )  
*page 139*    virtual long e\_user( long id, void \*data )  
*page 140*    XVT\_Control \*GetCtl( long cid )  
*page 140*    long GetCtlCount()  
*page 141*    EVENT\_MASK GetEventMask() const  
*page 141*    XVT\_Control \*GetFirstCtl()  
*page 142*    XVT\_ChildBase \*GetFirstWin()  
*page 142*    XVT\_Control \*GetNextCtl()  
*page 143*    XVT\_ChildBase \*GetNextWin()  
*page 143*    long GetWinCount()  
*page 144*    void Invalidate()  
*page 144*    void Invalidate( XVT\_Rctregion )  
*page 145*    void Scroll(  
                   XVT\_Rct boundary,  
                   long dh,  
                   long dv )  
*page 146*    void SetEventMask( EVENT\_MASK ask )  
*page 148*    void SetInnerRect( XVT\_Rct r )

### **From XVT\_Base**

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()

<i>page 11</i>	<code>virtual XVT_ChildWin *CastToChildWin()</code>
<i>page 11</i>	<code>virtual XVT_DetachedWin *CastToDetachedWin()</code>
<i>page 11</i>	<code>virtual XVT_Dialog *CastToDialog()</code>
<i>page 11</i>	<code>virtual XVT_DrawableContainer*CastToDrawableContainer()</code>
<i>page 11</i>	<code>virtual XVT_Edit *CastToEdit()</code>
<i>page 11</i>	<code>virtual XVT_GroupBox *CastToGroupBox()</code>
<i>page 11</i>	<code>virtual XVT_Icon *CastToIcon()</code>
<i>page 11</i>	<code>virtual XVT_ListBox *CastToListBox()</code>
<i>page 11</i>	<code>virtual XVT_ListButton *CastToListButton()</code>
<i>page 11</i>	<code>virtual XVT_ListEdit *CastToListEdit()</code>
<i>page 11</i>	<code>virtual XVT_MenuWin *CastToMenuWin()</code>
<i>page 11</i>	<code>virtual XVT_PrintWin *CastToPrintWin()</code>
<i>page 11</i>	<code>virtual XVT_RadioButton *CastToRadioButton()</code>
<i>page 11</i>	<code>virtual XVT_ScreenWin *CastToScreenWin()</code>
<i>page 11</i>	<code>virtual XVT_ScrollBar *CastToScrollBar()</code>
<i>page 11</i>	<code>virtual XVT_StaticText *CastToStaticText()</code>
<i>page 11</i>	<code>virtual XVT_TaskWin *CastToTaskWin()</code>
<i>page 11</i>	<code>virtual XVT_TopLevelWin *CastToTopLevelWin()</code>
<i>page 12</i>	<code>virtual XVT_Rct GetInnerRect()</code>
<i>page 13</i>	<code>virtual XVT_Rct GetOuterRect()</code>



# Member Functions

---

## MenuItem::check

CHECK OR UNCHECK A MENU ITEM

---

### Prototypes

```
void  
check(  
    BOOLEAN  
        check_me = TRUE )
```

### Parameters

check\_me  
A flag that is TRUE if the menu item is to be checked, FALSE if unchecked.

### Description

Checks or unchecks a menu item.

### Equivalent C Function

win\_menu\_check()

---

## MenuItem::disable

DISABLE A MENU ITEM

---

### Prototypes

```
void  
disable()
```

### Description

Disables a menu item.

### Equivalent C Function

win\_menu\_enable()

---

## MenuItem::enable

ENABLE OR DISABLE A MENU ITEM

---

### Prototypes

```
void  
enable(  
    BOOLEAN                                enable_me = TRUE )
```

### Parameters

enable\_me  
A flag that is TRUE if the menu item is to be enabled, FALSE if disabled.

### Description

Enables or disables a menu item.

### Equivalent C Function

```
win_menu_enable()
```

---

## MenuItem::tag

RETRIEVE THE TAG OF THE ASSOCIATED MENU ITEM

---

### Prototypes

```
MENU_TAG  
tag()
```

### Return Value

The tag of the associated menu item.

---

## MenuItem::uncheck

UNCHECK A MENU ITEM

---

### Prototypes

```
void  
uncheck()
```

### Description

Unchecks a menu item.

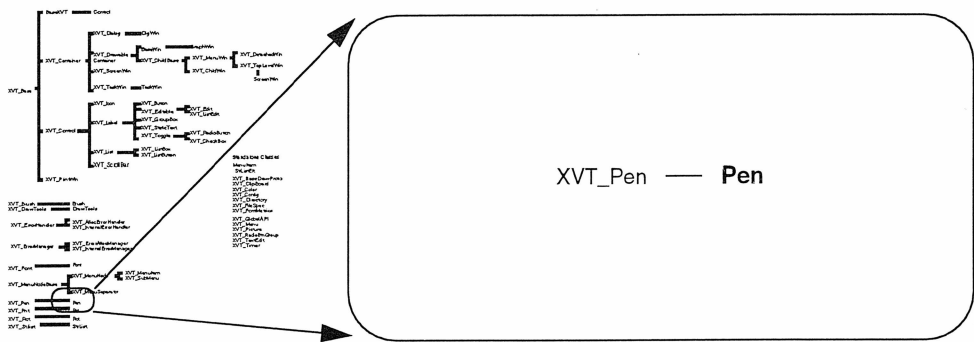
### Equivalent C Function

win\_menu\_check()

## Implementation Members

class\_name

# Pen



## Overview

Header File	ktool.hpp
Source File	ktool.cc
Superclass	XVT_Pen
Subclasses	
Usage	Concrete

Instances of this class define how line drawing primitives are rendered in a window. Each window maintains an instance of this class.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use XVT\_Pen instead.

## Constructors

```
Pen( COLOR clr = COLOR_BLACK, PAT_STYLE pat = PAT_SOLID,
    short wdh = 1 )
```

# Member Functions

## Pen::color

---

GET OR SET THE PEN'S COLOR

---

### Prototypes

```
COLOR
color()

void
color(      COLOR                clr )
```

### Parameters

clr  
The pen's new color.

### Return Value

color()  
The pen's color.

### Description

color(clr)  
Sets the pen's color.

---

## Pen::pat

---

GET OR SET THE PEN'S PATTERN

---

### Prototypes

```
PAT_STYLE
pat()

void
pat(      PAT_STYLE                p )
```

### Parameters

p  
The pen's new pattern.



**Return Value**

pat()  
The pen's pattern.

**Description**

pat( p )  
Sets the pen's pattern.

---

**Pen::width**

GET OR SET A PEN'S WIDTH

---

**Prototypes**

```
int
width()

void
width(
    short                wdth )
```

**Parameters**

wdth  
The pen's new width.

**Return Value**

width()  
The pen's width.

**Description**

width( short wdth )  
Sets the pen's width.

**Implementation Members**

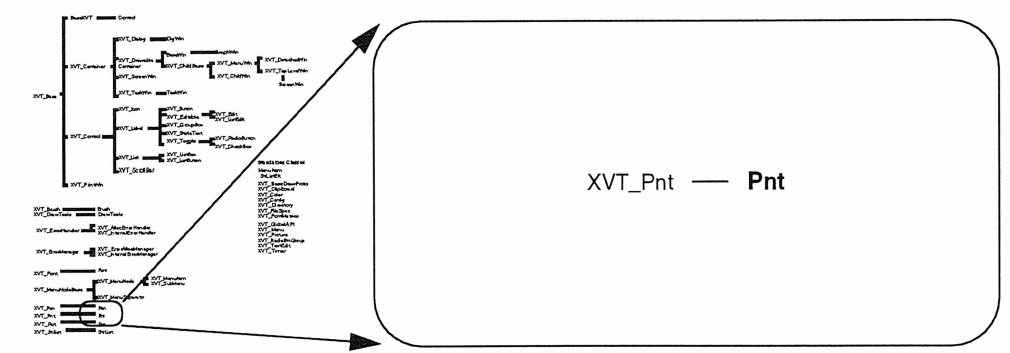
class\_name

**Inherited Member Functions****From XVT\_Pen**

page 296    XVT\_Color GetColor()  
page 296    PAT\_STYLE GetPattern()  
page 296    PEN\_STYLE GetStyle()

<i>page 297</i>	<code>short GetWidth()</code>
<i>page 297</i>	<code>void SetColor( XVT_Color c )</code>
<i>page 297</i>	<code>void SetPattern( PAT_STYLE p )</code>
<i>page 298</i>	<code>void SetStyle( PEN_STYLE s )</code>
<i>page 299</i>	<code>void SetWidth( short w )</code>

# Pnt



## Overview

Header File	kpnt.hpp
Source File	
Superclass	XVT_Pnt
Subclasses	
Usage	Concrete

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use XVT\_Pnt instead.

## Constructors

Pnt( int x = 0, int y = 0 )

# Member Functions

---

## Pnt::set

SET A POINT'S X AND Y COORDINATES

---

### Prototypes

```
Pnt  
set(  
    int  
    int  
    x = 0,  
    y = 0 )
```

### Parameters

x  
The point's new X coordinate.

y  
The point's new Y coordinate.

### Return Value

The point.

---

## Pnt::x

SET OR RETRIEVE THE POINT'S X COORDINATE

---

### Prototypes

```
short  
x()  
  
void  
x(  
    int  
    xx )
```

### Parameters

xx  
The point's new X coordinate.

**Return Value**

x()  
The point's current X coordinate.

**Description**

x(xx)  
Sets the point's X coordinate.

---

**Pnt::y**

SET OR RETRIEVE THE POINT'S Y COORDINATE

---

**Prototypes**

```
short
y()
void
y(
    int yy )
```

**Parameters**

yy  
The point's new Y coordinate.

**Return Value**

y()  
The point's current Y coordinate.

**Description**

y(yy)  
Set the point's Y coordinate

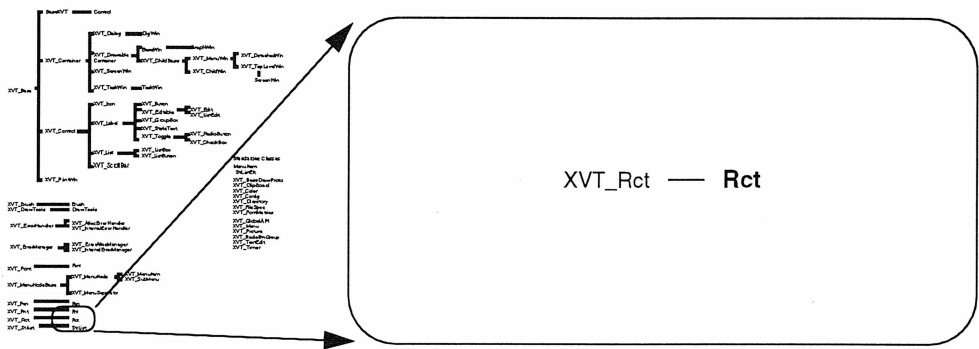
**Implementation Members**

class\_name

**Inherited Member Functions****From XVT\_Pnt**

page 307    short GetX( void )  
 page 307    short GetY( void )  
 page 308    void SetX( short pos )  
 page 308    void SetY( short pos )

# Rct



## Overview

Header File	krct.hpp
Source File	
Superclass	XVT_Rct
Subclasses	
Usage	Concrete

Instances of this class model mathematical rectangles. This class is completely compatible with the XVT++ 1.1 class of the same name. It is included for backwards compatibility only. New applications should use the XVT\_Rct class.

## Constructors

```
Rct( int left, int top, int right, int bottom )  
Rct( Pnt ul, Pnt lr )  
Rct()
```

# Member Functions

---

## Rct::bottom

RETRIEVE THE BOTTOM EDGE OF THE RECTANGLE

---

### Prototypes

```
int  
bottom()
```

### Return Value

The bottom edge of the rectangle.

---

## Rct::empty

DETERMINE IF A RECTANGLE IS EMPTY

---

### Prototypes

```
BOOLEAN  
empty()
```

### Return Value

A flag that is TRUE if the rectangle is empty, FALSE otherwise.

### Equivalent C Function

```
is_rect_empty()
```

---

## Rct::left

RETRIEVE THE LEFT EDGE OF THE RECTANGLE

---

### Prototypes

```
int  
left()
```

### Return Value

The left edge of the rectangle.

---

## Rct::right

RETRIEVE THE RIGHT EDGE OF THE RECTANGLE

---

### Prototypes

```
int
right()
```

### Return Value

The right edge of the rectangle.

---

## Rct::set

SET A RECTANGLE'S DIMENSIONS

---

### Prototypes

```
Rct
set(
    int          left = 0,
    int          top  = 0,
    int          right = 0,
    int          bottom = 0 )

Rct
set(
    Pnt          ul,
    Pnt          lr )
```

### Parameters

```
left
    The left edge of the rectangle.

top
    The top edge of the rectangle.

right
    The right edge of the rectangle.

bottom
    The bottom edge of the rectangle.

ul
    The upper-left corner point of the rectangle.

lr
    The lower-right corner point of the rectangle.
```



**Return Value**

A copy of the rectangle.

**Description**

Sets a rectangle's dimensions.

**Equivalent C Function**

set\_rect()

set\_rect\_empty()

---

**Rct::top**

RETRIEVE THE TOP EDGE OF THE RECTANGLE

---

**Prototypes**

```
int
top()
```

**Return Value**

The top edge of the rectangle.

**Implementation Members**

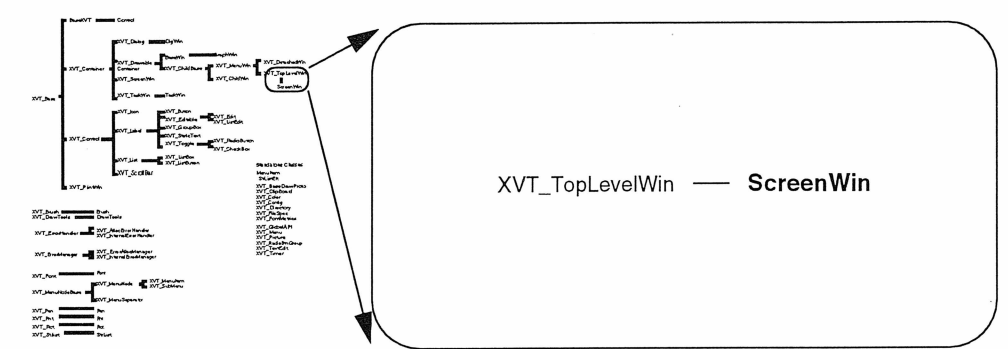
class\_name

**Inherited Member Functions****From XVT\_Rct**

- page 324*    XVT\_Pnt Constrain( XVT\_Pntpoint )
- page 325*    BOOLEAN Contains( XVT\_Pnt point ) const
- page 325*    BOOLEAN Contains( XVT\_Rct rect ) const
- page 326*    short Difference( XVT\_Rct& boundary, XVT\_Rct \*list ) const
- page 326*    virtual XVT\_Pnt GetBottomLeft() const
- page 327*    virtual XVT\_Pnt GetBottomRight() const
- page 327*    XVT\_Pnt GetDimVect() const

*page 327*     virtual XVT\_Pnt GetTopLeft() const  
*page 328*     virtual XVT\_Pnt GetTopRight() const  
*page 328*     short Height() const  
*page 328*     BOOLEAN Intersect( XVT\_Rct& boundary ) const  
*page 329*     BOOLEAN IsEmpty() const  
*page 329*     XVT\_Rct Normalize() const  
*page 330*     void SetBottomLeft( XVT\_Pnt point )  
*page 330*     void SetBottomRight( XVT\_Pnt point )  
*page 330*     void SetTopLeft( XVT\_Pnt point )  
*page 331*     void SetTopRight( XVT\_Pnt point )  
*page 331*     XVT\_Pnt TransToGlobal( XVT\_Pnt point ) const  
*page 332*     XVT\_Pnt TransToLocal( XVT\_Pnt point ) const  
*page 332*     short Width() const

# ScreenWin



## Overview

Header File	kscreen.hpp
Source File	kscreen.cc
Superclass	XVT_TopLevelWin
Subclasses	
Usage	Abstract

In XVT++ 1.1, the `ScreenWin` class defined the interface to all non-task windows.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use either `XVT_TopLevelwin` or `XVT_DetachedWin` instead.

You use this class by creating a subclass that overrides virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

```
ScreenWin()
virtual ~ScreenWin()
```

## Member Functions

The following functions are identical to those implemented by BaseXVT:

```
page 425    virtual void disable()
page 425    virtual void enable( BOOLEAN enabled = TRUE )
page 426    WIN_DEF* get_def() const
page 426    Rct get_rect() const
page 427    virtual SSTR* get_text( char* buffer, int len ) const
page 427    WIN_TYPE get_type() const
page 428    virtual void hide()
page 428    virtual void move( Rct boundary )
page 428    WINDOW parent()
page 429    void put_def( WIN_DEF* in_def )
page 429    virtual void set_text( char* str )
page 430    virtual void show( BOOLEAN visible = TRUE )
```

The following functions are identical to those implemented by BaseWin:

```
page 415    long dispatch( EVENT* event)
page 416    virtual void e_command( MenuItem mi, BOOLEAN shift,
    BOOLEAN control )
page 417    virtual void e_control( int cid, CONTROL_INFO* info )
page 418    virtual void e_deactivate()
page 416    virtual void e_activate()
page 418    Rct get_client()
page 419    EVENT_MASK get_mask()
page 419    WINDOW get_win()
page 420    void set_mask( EVENT_MASK mask )
```

page 420    long set\_timer( long interval )

The following functions are identical to those implemented by GraphWin:

page 469    void arc( Rct lrct, Pnt start, Pnt stop )

page 470    void clear()

page 470    DrawTools get\_tools() const

page 470    void icon( Pnt p, int rid )

page 471    void line( Pnt from, Pnt to, BOOLEAN start\_arrow = FALSE, BOOLEAN end\_arrow = FALSE )

page 472    void move\_to( Pnt p )

page 472    void oval( Rct r )

page 473    void pie( Rctr, Pnt start, Pnt stop )

page 473    void polygon( Pnt\* points, int npoints )

page 474    void polyline( Pnt\* points, int npoints )

page 474    void rectangle( Rct r )

page 475    void rounded\_rectangle( Rct r, int oval\_width, int oval\_height )

page 475    void set\_brush( Brush b )

page 476    void set\_font( Font font )

page 476    void set\_mode( DRAW\_MODE mode )

page 477    void set\_pen( Pen p )

page 477    void set\_tools( DrawTools tools )

page 478    void text( Pnt p, SSTR\* str, int i = -1 )

---

## ScreenWin::create

CREATE A WINDOW FROM RESOURCES

---

### Prototypes

```
virtual BOOLEAN
create(
    int                rid,
    long               appdata = 0L )
```

**Parameters**

`rid`  
The window resource ID.

`appdata`  
The application data associated with this window.

**Return Value**

A flag that is TRUE if the operation succeeded, FALSE if it failed.

**Description**

Creates a window from resources.

**Equivalent C Function**

`create_res_window()`

---

## ScreenWin::create\_def

CREATE WINDOW FROM A WIN\_DEF

---

**Prototypes**

```
virtual BOOLEAN  
create_def(  
    long                                appdata = 0L )
```

**Parameters**

`appdata`  
The application data associated with this window.

**Return Value**

A flag that is TRUE if the operation succeeded, FALSE if it failed.

**Description**

Creates a window from the stored WIN\_DEF structure.

**Equivalent C Function**

`create_def_window()`

## ScreenWin::create\_scratch

CREATE A WINDOW FROM PARAMETERS

### Prototypes

```
virtual BOOLEAN
create_scratch(
    Rct      lrct,
    int      menu_rid,
    WIN_TYPE wtype = W_DOC,
    SSTR*    title = "",
    long     win_flags = WSF_SIZE | WSF_CLOSE,
    long     appdata = 0L )
```

### Parameters

**lrct**  
The new window's client area.

**menu\_rid**  
The resource ID of the new window's menu.

**wtype**  
The type of the new window.

**title**  
The new window's title.

**win\_flags**  
The new window's attribute flags.

**appdata**  
The new window's application data.

### Return Value

A flag that is TRUE if the operation succeeded, FALSE if it failed.

### Description

Creates a window specified by parameters.

### Equivalent C Function

create\_window()

---

## ScreenWin::get\_metrics

RETRIEVE THE CURRENT FONT METRICS

---

### Prototypes

```
virtual void  
get_metrics(  
    int*          leading,  
    int*          ascent,  
    int*          descent )
```

### Parameters

leading  
The leading of the current font.

ascent  
The ascent of the current font.

descent  
The descent of the current font.

### Description

Retrieves the current font metrics.

### Equivalent C Function

win\_get\_font\_metrics()

---

## ScreenWin::set\_def

SET THE ASSOCIATED WIN\_DEF FROM A RESOURCE

---

### Prototypes

```
virtual BOOLEAN  
set_def(  
    int          rid )
```

### Parameters

rid  
The resource ID from which to get the window definition.

### Return Value

A flag that is TRUE if the operation succeeded, FALSE if it failed.



**Description**

Sets the associated WIN\_DEF from a resource.

**Equivalent C Function**

get\_res\_window()

**Implementation Members**

class\_name

**Inherited Member Functions****From XVT\_TopLevelWin**

*page 401*    `BOOLEAN Init(  
            WIN_TYPE wtype,  
            XVT_Rct boundary,  
            char* title,  
            long menu_rid,  
            long flags )`

*page 401*    `BOOLEAN Init( long rid )`

**From XVT\_MenuWin**

*page 286*    `virtual void e_close()`

*page 287*    `virtual void e_font( XVT_Font font, FONT_PART part )`

*page 287*    `XVT_Menu *GetMenu()`

*page 288*    `void GetTitle( char *buffer, long len )`

*page 289*    `void SetDocTitle( char *str )`

*page 289*    `void SetFontMenu( XVT_Font font )`

*page 290*    `void SetMenu( XVT_Menu *menu )`

*page 291*    `void SetTitle( char *str )`

**From XVT\_ChildBase**

*page 49*    `virtual void e_hscroll( SCROLL_CONTROL activity, short  
                         pos )`

*page 49*    `virtual void e_vscroll( SCROLL_CONTROL activity, short  
                         pos )`

*page 50*    `XVT_TextEdit* GetActiveTextEdit()`

<i>page 50</i>	XVT_Pnt GetCaretPos() const
<i>page 51</i>	BOOLEAN GetCaretState() const
<i>page 51</i>	BOOLEAN GetEnabledState()
<i>page 51</i>	XVT_ChildBase *GetParent() const
<i>page 52</i>	long GetScrollPosition( SCROLL_TYPE scroll_type ) const
<i>page 52</i>	long GetScrollProportion( SCROLL_TYPE scroll_type ) const
<i>page 53</i>	void GetScrollRange( SCROLL_TYPE scroll_type, long *min, long *max ) const
<i>page 54</i>	XVT_TextEdit* GetTextEdit( long id )
<i>page 54</i>	BOOLEAN GetVisibleState()
<i>page 55</i>	void MakeFront()
<i>page 55</i>	void ReleaseMouse()
<i>page 56</i>	void SetCaretDimensions( XVT_Pnt vector )
<i>page 56</i>	void SetCaretPos( XVT_Pnt point )
<i>page 57</i>	void SetCaretState( BOOLEAN state )
<i>page 57</i>	void SetCursor( CURSOR cursor )
<i>page 58</i>	void SetEnabledState( BOOLEAN state )
<i>page 59</i>	void SetScrollPosition( SCROLL_TYPE scroll_type, long position )
<i>page 60</i>	void SetScrollProportion( SCROLL_TYPE scroll_type, long proportion )
<i>page 60</i>	void SetScrollRange( SCROLL_TYPE scroll_type, long min, long max, long pos )
<i>page 61</i>	void SetVisibleState( BOOLEAN f )
<i>page 62</i>	void TrapMouse()

### **From XVT\_DrawableContainer**

<i>page 129</i>	void Clear()
<i>page 129</i>	void Clear( XVT_Color color )
<i>page 129</i>	void Close()
<i>page 128</i>	XVT_BaseDrawProto* DrawProtocol

<i>page 130</i>	virtual void e_char( short chr, BOOLEAN shift, BOOLEAN control)
<i>page 131</i>	virtual void e_create()
<i>page 132</i>	virtual void e_destroy()
<i>page 132</i>	virtual void e_focus( BOOLEAN active )
<i>page 133</i>	virtual void e_mouse_dbl( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 134</i>	virtual void e_mouse_down( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_move( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 135</i>	virtual void e_mouse_up( XVT_Pnt point, BOOLEAN shift, BOOLEAN control, short button )
<i>page 136</i>	virtual void e_size( XVT_Rct boundary )
<i>page 137</i>	virtual void e_timer( long id )
<i>page 137</i>	virtual void e_update( XVT_Rct boundary )
<i>page 139</i>	virtual long e_user( long id, void *data )
<i>page 140</i>	XVT_Control *GetCtl( long cid )
<i>page 140</i>	long GetCtlCount()
<i>page 141</i>	EVENT_MASK GetEventMask() const
<i>page 141</i>	XVT_Control *GetFirstCtl()
<i>page 142</i>	XVT_ChildBase *GetFirstWin()
<i>page 142</i>	XVT_Control *GetNextCtl()
<i>page 143</i>	XVT_ChildBase *GetNextWin()

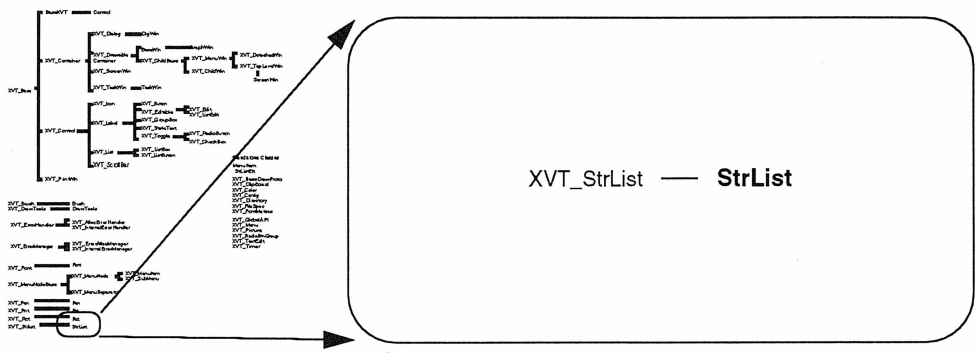
*page 143*    long GetWinCount()  
*page 144*    void Invalidate()  
*page 144*    void Invalidate( XVT\_Rctregion )  
*page 145*    void Scroll(  
              XVT\_Rctboundary,  
              long dh,  
              long dv )  
*page 146*    void SetEventMask( EVENT\_MASK ask )  
*page 148*    void SetInnerRect( XVT\_Rct r )

### From XVT\_Base

*page 11*    virtual BaseWin\* CastToBaseWin()  
*page 10*    virtual DlgWin\* CastToDlgWin()  
*page 10*    virtual ScreenWin\* CastToScreenWin11()  
*page 10*    virtual TaskWin\* CastToTaskWin11()  
*page 11*    virtual XVT\_Button \*CastToButton()  
*page 11*    virtual XVT\_CheckBox \*CastToCheckBox()  
*page 11*    virtual XVT\_ChildWin \*CastToChildWin()  
*page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()  
*page 11*    virtual XVT\_Dialog \*CastToDialog()  
*page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()  
*page 11*    virtual XVT\_Edit \*CastToEdit()  
*page 11*    virtual XVT\_GroupBox \*CastToGroupBox()  
*page 11*    virtual XVT\_Icon \*CastToIcon()  
*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()

*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

# StrList



## Overview

Header File	kslist.hpp
Source File	kslist.cc
Superclass	XVT_StrList
Subclasses	
Usage	Concrete

Instances of this class represent lists of strings.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is included for backwards compatibility only. New programs should use `XVT_StrList` instead.

## Constructors

```
StrList( SLIST x = NULL )
~StrList()
```

# Member Functions

## StrList::add

ADD AN ELEMENT OR ELEMENTS TO THE LIST

### Prototypes

```

BOOLEAN
add(
    StrListElt    e,
    char*         sx,
    long          data = 0 )

BOOLEAN
add(
    char*         sx,
    long          data = 0 )

```

### Parameters

**e**  
The element before which to add the new element(s).

**sx**  
Either a character string or an (SLIST\*).

**data**  
The data portion of the new element.

**unique**  
A flag that is TRUE if only unique elements are to be added to the list, FALSE if not.

**case\_sensitive**  
A flag that is TRUE if element comparisons are to be case-sensitive, FALSE if case is to be ignored.

### Return Value

A flag that is TRUE if the operation succeeded, FALSE if it failed.

### Description

Adds an element or elements to the list.

```

add( e, sx, data )
    Adds an element or elements to the list immediately in front of
    the element e.

add( sx, data )
    Adds an element or elements to the end of the list.

```

**Equivalent C Function**

slist\_add()

---

**StrList::add\_sorted**ADD AN ELEMENT TO A STRING LIST IN ORDER

---

**Prototypes**

```

void
add_sorted(
    char*      str,
    long       data = 0L,
    BOOLEAN    unique = FALSE,
    BOOLEAN    case_sensitive = FALSE )

```

**Parameters**

**str**  
The string portion of the element to add.

**data**  
The data portion of the element to add.

**unique**  
A flag that is TRUE if duplicate elements are not to be added to the string list, FALSE if they are.

**case\_sensitive**  
A flag that is TRUE if element comparisons are to be case-sensitive, FALSE if they are to ignore case.

**Description**

Adds an element to a string list in lexicographic order.

---

**StrList::count**RETRIEVE THE NUMBER OF ELEMENTS IN A LIST

---

**Prototypes**

```

int
count()

```

**Return Value**

The number of items in the list.



**Equivalent C Function**`slist_count()`

---

**StrList::dbg**

DUMP A STRING LIST TO THE DEBUG FILE

---

**Prototypes**

```
void  
dbg()
```

**Description**

Dumps a string list to the debug file.

**Equivalent C Function**`slist_dbg()`

---

**StrList::elt**

RETRIEVE THE CONTENTS OF A STRING LIST ELEMENT

---

**Prototypes**

```
char*  
elt(  
    int          index,  
    long*        datap = NULL )
```

**Parameters**

`index`  
The index of the element to retrieve.

`datap`  
Storage to receive the element data.

**Return Value**

The string portion of the element.

**Equivalent C Function**`slist_elt()`

---

## StrList::first

BEGIN A TRAVERSAL OF THE STRING LIST

---

### Prototypes

```
StrListElt  
first()
```

### Return Value

The first element in the list.

### Description

Begins a traversal of the string list.

### Equivalent C Function

```
slist_first()
```

---

## StrList::get

RETRIEVE AN ELEMENT IN A LIST

---

### Prototypes

```
char*  
get(  
    StrListElt  
    long*  
    e,  
    datap = NULL )
```

### Parameters

e  
The element to retrieve.

datap  
Storage to receive the element data.

### Return Value

The string portion of the element.

### Description

Retrieves an element in a list.

### Equivalent C Function

```
slist_get()
```

---

## StrList::next

RETRIEVE THE NEXT ELEMENT IN A STRING LIST

---

### Prototypes

```
StrListElt  
next(  
    StrListElt    e )
```

### Parameters

e  
The list element.

### Return Value

The list element, e.

### Description

Retrieves the next element in a string list.

### Equivalent C Function

slist\_next()

---

## StrList::rem

REMOVE AN ELEMENT FROM A STRING LIST

---

### Prototypes

```
BOOLEAN  
rem(  
    StrListElt    e )
```

### Parameters

e  
The element to be removed.

### Return Value

A flag that is TRUE if the operation succeeded, FALSE if not.

### Description

Removes an element from a string list.

**Equivalent C Function**

slist\_rem()

---

**StrList::valid**DETERMINE IF A STRING LIST IS VALID

---

**Prototypes**BOOLEAN  
valid()**Return Value**

A flag that is TRUE if the list is a valid list, FALSE if not.

**Description**

Determines if a string list is valid.

**Equivalent C Function**

slist\_valid()

**Implementation Members**

class\_name

**Inherited Member Functions****From XVT\_StrList**

*page 350*    void Add( long element, char\* str, long data = 0L )

*page 350*    void Add( long element, XVT\_StrList\* sl )

*page 350*    void Add( char\* ch, long data = 0L )

*page 350*    void Add( XVT\_StrList\* sl )

*page 351*    void AddSorted( char\* str, long data = 0L, BOOLEAN unique  
                  = FALSE, BOOLEAN case\_sensitive = FALSE )

*page 352*    long Count()

*page 352*    void Debug()

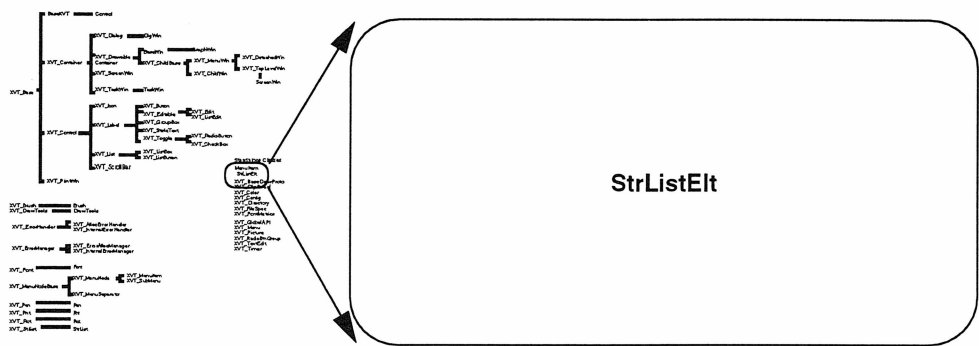
*page 353*    `BOOLEAN GetElement( long index, const char** str, long* data ) const;`

*page 353*    `BOOLEAN GetFirst( const char** str, long* data, long* index = 0 )`

*page 354*    `BOOLEAN GetNext( const char** str, long* data )`

*page 355*    `void Remove( long index )`

# StrListElt



## Overview

Header File	kslist.hpp
Source File	kslist.cc
Superclass	
Subclasses	
Usage	Concrete

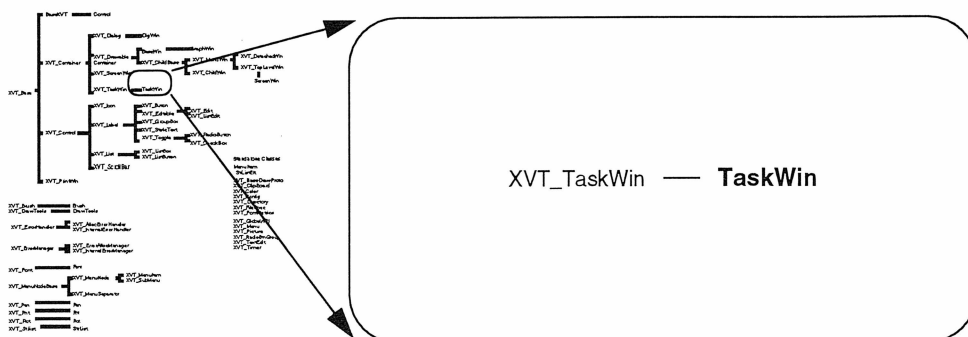
This class simply defines an opaque context object that gives a location in a string list. The application developer never instantiates this class; instances are created as needed by the StrList class.

This class is completely compatible with the XVT++ 1.1 class of the same name. It is included for backwards compatibility only. New applications should use XVT\_StrList, which does not expose a context class.

## Implementation Members

class\_name

# TaskWin



## Overview

<b>Header File</b>	ktask.hpp
<b>Source File</b>	ktask.cc
<b>Superclass</b>	XVT_TaskWin
<b>Subclasses</b>	
<b>Usage</b>	Abstract

The TaskWin class defines the interface to the task window.

This class is completely compatible with the XVT++ 1.1 class of the same name. This class is provided for backwards compatibility only. For new applications, we recommend that you use XVT\_TaskWin instead.

You use this class by creating a subclass that overrides the virtual event handling member functions with implementations that actually do something in response to events.

## Constructors

```
TaskWin()
~TaskWin()
```

## Member Functions

The following functions are identical to those implemented by BaseXVT:

<i>page 425</i>	<code>virtual void disable()</code>
<i>page 425</i>	<code>virtual void enable( BOOLEAN enabled = TRUE )</code>
<i>page 426</i>	<code>WIN_DEF* get_def() const</code>
<i>page 426</i>	<code>Rct get_rect() const</code>
<i>page 427</i>	<code>virtual SSTR* get_text( char* buffer, int len )</code>
<i>page 427</i>	<code>WIN_TYPE get_type() const</code>
<i>page 428</i>	<code>virtual void hide()</code>
<i>page 428</i>	<code>virtual void move( Rct boundary )</code>
<i>page 428</i>	<code>WINDOW parent()</code>
<i>page 429</i>	<code>void put_def( WIN_DEF* In_def )</code>
<i>page 429</i>	<code>virtual void set_text( char* str)</code>
<i>page 430</i>	<code>virtual void show( BOOLEAN visible = TRUE )</code>

The following functions are identical to those implemented by BaseWin:

<i>page 416</i>	<code>virtual void e_activate()</code>
<i>page 416</i>	<code>virtual void e_command( MenuItem mi, BOOLEAN shift, BOOLEAN control )</code>
<i>page 417</i>	<code>virtual void e_control( int cid, CONTROL_INFO* info )</code>
<i>page 418</i>	<code>virtual void e_deactivate()</code>
<i>page 415</i>	<code>long dispatch( EVENT* event)</code>
<i>page 418</i>	<code>Rct get_client() const</code>
<i>page 419</i>	<code>EVENT_MASK get_mask() const</code>
<i>page 419</i>	<code>WINDOW get_win() const</code>
<i>page 419</i>	<code>void set_font( Font, BOOLEAN )</code>
<i>page 420</i>	<code>void set_mask( EVENT_MASK mask )</code>
<i>page 420</i>	<code>long set_timer( long interval )</code>



---

## TaskWin:: begin

START AN XVT++ APPLICATION

---

### Prototypes

```
virtual void  
begin(  
    int          argc,  
    char*        argv[],  
    long         flags = 0L,  
    XVT_CONFIG*  config = NULL )
```

### Parameters

**argc**  
The number of entries in the argv array.

**argv**  
The array of argument words, null terminated.

**flags**  
Not used.

**config**  
Application configuration data.

### Return Value

Never returns.

### Description

Starts an XVT++ application.

### Equivalent C Function

xvt\_system()

---

## TaskWin:: get\_config

RETRIEVE APPLICATION CONFIGURATION DATA

---

### Prototypes

```
virtual XVT_CONFIG*  
get_config() const
```

**Return Value**

The configuration data passed into begin.

**Description**

Retrieves application configuration data.

## Inherited Member Functions

**From XVT\_TaskWin**

- page 362*    void Close()
- page 363*    virtual void e\_close()
- page 363*    virtual void e\_create()
- page 364*    virtual void e\_destroy()
- page 364*    virtual BOOLEAN e\_quit( BOOLEAN query\_only )
- page 366*    virtual void Init( int argc, char \*argv[], unsigned long flags, XVT\_Config config )
- page 367*    virtual BOOLEAN QuitOK()

**From XVT\_Base**

- page 11*    virtual BaseWin\* CastToBaseWin()
- page 10*    virtual DlgWin\* CastToDlgWin()
- page 10*    virtual ScreenWin\* CastToScreenWin11()
- page 10*    virtual TaskWin\* CastToTaskWin11()
- page 11*    virtual XVT\_Button \*CastToButton()
- page 11*    virtual XVT\_CheckBox \*CastToCheckBox()
- page 11*    virtual XVT\_ChildWin \*CastToChildWin()
- page 11*    virtual XVT\_DetachedWin \*CastToDetachedWin()
- page 11*    virtual XVT\_Dialog \*CastToDialog()
- page 11*    virtual XVT\_DrawableContainer\*CastToDrawableContainer()
- page 11*    virtual XVT\_Edit \*CastToEdit()
- page 11*    virtual XVT\_GroupBox \*CastToGroupBox()
- page 11*    virtual XVT\_Icon \*CastToIcon()

*page 11*    virtual XVT\_ListBox \*CastToListBox()  
*page 11*    virtual XVT\_ListButton \*CastToListButton()  
*page 11*    virtual XVT\_ListEdit \*CastToListEdit()  
*page 11*    virtual XVT\_MenuWin \*CastToMenuWin()  
*page 11*    virtual XVT\_PrintWin \*CastToPrintWin()  
*page 11*    virtual XVT\_RadioButton \*CastToRadioButton()  
*page 11*    virtual XVT\_ScreenWin \*CastToScreenWin()  
*page 11*    virtual XVT\_ScrollBar \*CastToScrollBar()  
*page 11*    virtual XVT\_StaticText \*CastToStaticText()  
*page 11*    virtual XVT\_TaskWin \*CastToTaskWin()  
*page 11*    virtual XVT\_TopLevelWin \*CastToTopLevelWin()  
*page 12*    virtual XVT\_Rct GetInnerRect()  
*page 13*    virtual XVT\_Rct GetOuterRect()

C→C++  
Index

# XVT++

## INDEX OF EQUIVALENT C FUNCTIONS

### A

about\_box()  
     XVT\_GlobalAPI::About, 183

### C

caret\_off()  
     XVT\_ChildBase::SetCaretState, 57  
 caret\_on()  
     XVT\_ChildBase::SetCaretPos, 57  
     XVT\_ChildBase::SetCaretState, 57  
 cb\_close()  
     XVT\_ClipBoard::GetData, 77  
     XVT\_ClipBoard::PutData, 79  
 cb\_format\_avail()  
     XVT\_ClipBoard::FormatAvail, 76  
 cb\_free()  
     XVT\_ClipBoard::GetData, 77  
     XVT\_ClipBoard::PutData, 79  
 cb\_get()  
     XVT\_ClipBoard::GetData, 77  
 cb\_malloc  
     (XVT\_ClipBoard::GetData, 77  
 cb\_malloc()  
     XVT\_ClipBoard::PutData, 79  
 cb\_open()  
     XVT\_ClipBoard::GetData, 77  
     XVT\_ClipBoard::PutData, 79  
 cb\_put()  
     XVT\_ClipBoard::PutData, 79

chg\_dir()  
     XVT\_GlobalAPI::ChgDir, 185  
 clear\_window()  
     XVT\_DrawableContainer::Clear, 129  
 close\_window()  
     XVT\_Control::Close, 93  
     XVT\_Dialog::Close, 109  
     XVT\_DrawableContainer::Close, 130  
     XVT\_TaskWin::Close, 363  
 create\_control()  
     XVT\_Control::Init, 96  
     XVT\_Icon::Init, 231  
     XVT\_Label::Init, 240  
     XVT\_ScrollBar::Init, 341  
 create\_def\_control()  
     XVT\_Control::Init, 96  
     XVT\_Icon::Init, 231  
     XVT\_Label::Init, 240  
     XVT\_ScrollBar::Init, 341  
 create\_def\_dialog()  
     XVT\_Dialog::Init, 119  
 create\_def\_window()  
     XVT\_ChildWin::Init, 68  
     XVT\_DetachedWin::Init, 103  
     XVT\_TopLevelWin::Init, 403  
 create\_res\_dialog()  
     XVT\_Dialog::Init, 119

- create\_res\_window()
  - XVT\_ChildWin::Init, 68
  - XVT\_DetachedWin::Init, 103
  - XVT\_TopLevelWin::Init, 403
- create\_window()
  - XVT\_ChildWin::Init, 68
  - XVT\_DetachedWin::Init, 103
  - XVT\_TopLevelWin::Init, 403
- D**
- dbg2()
  - XVT\_GlobalAPI::Debug2, 186
- dir\_to\_str()
  - XVT\_Directory::DirToStr, 126
- E**
- enable\_window()
  - XVT\_ChildBase::SetEnabledState, 59
  - XVT\_Control::SetEnabledState, 97
  - XVT\_Dialog::SetEnabledState, 120
- F**
- find\_eol()
  - XVT\_GlobalAPI::FindEOL, 188
- G**
- galloc()
  - XVT\_GlobalAPI::GAlloc, 190
- get\_client\_rect()
  - XVT\_Base::GetInnerRect, 12
- get\_clip()
  - XVT\_BaseDrawProto::GetClip, 26
- get\_ctl\_window()
  - XVT\_Control::GetID, 95
- get\_default\_dir()
  - XVT\_GlobalAPI::GetDefaultDir, 192
- get\_dialog\_userdata()
  - XVT\_GlobalAPI::GetDialogUserData, 193
- get\_dir()
  - XVT\_GlobalAPI::GetDir, 194
- get\_event\_mask()
  - XVT\_Dialog::GetEventMask, 116
  - XVT\_DrawableContainer::GetEventMask, 141
- get\_front\_top\_level\_window()
  - XVT\_GlobalAPI::GetDefaultBackColor, 191
  - XVT\_GlobalAPI::GetFrontTopLevelWin, 191
- get\_front\_window()
  - XVT\_GlobalAPI::GetFrontWin, 192
- get\_menu\_userdata()
  - XVT\_GlobalAPI::GetMenuUserData, 195
- get\_outer\_rect()
  - XVT\_Base::GetOuterRect, 13
- get\_parent()
  - XVT\_ChildBase::GetParent, 52
  - XVT\_Control::GetParent, 95
- get\_res\_str()
  - XVT\_GlobalAPI::GetResString, 195
- get\_scroll\_pos()
  - XVT\_ChildBase::GetScrollPosition, 52
- get\_scroll\_proportion()
  - XVT\_ChildBase::GetScrollProportion, 53
  - XVT\_ScrollBar::GetScrollProportion, 340
- get\_scroll\_range()
  - XVT\_ChildBase::GetScrollRange, 54
  - XVT\_ScrollBar::GetScrollPosition, 339
  - XVT\_ScrollBar::GetScrollRange, 340
- get\_str\_response()
  - XVT\_GlobalAPI::Response, 206
- get\_title()
  - XVT\_Dialog::GetTitle, 118
  - XVT\_Label::GetTitle, 239
  - XVT\_MenuWin::GetTitle, 288
- get\_tx\_edit
  - XVT\_ChildBase::GetTextEdit, 54
- get\_value( ATTR\_PS\_PRINT\_FILE\_NAME )
  - XVT\_PrintWin::GetOutputFile, 313
- get\_value()
  - XVT\_GlobalAPI::GetAttrValue, 191
- get\_window\_type()
  - XVT\_Base::GetType, 14
- get\_window\_userdata()
  - XVT\_GlobalAPI::GetWindowUserData, 196

gfree()  
     XVT\_GlobalAPI::GFree, 197  
glock()  
     XVT\_GlobalAPI::GLock, 197  
grealloc()  
     XVT\_GlobalAPI::GReAlloc, 198  
gsize()  
     XVT\_GlobalAPI::GSize, 199  
gunlock()  
     XVT\_GlobalAPI::GUnLock, 199

**I**

invalidate\_rect()  
     XVT\_DrawableContainer::Invalidate, 144  
is\_rect\_empty()  
     XVT\_Rct::IsEmpty, 329

**L**

list\_faces()  
     XVT\_GlobalAPI::ListFaces, 200  
list\_res\_str()  
     XVT\_GlobalAPI::ListResStrings, 201  
list\_windows()  
     XVT\_DrawableContainer::GetNextWin, 143  
     XVT\_ScreenWin::GetNextWin, 335

**M**

move\_window()  
     XVT\_Control::SetInnerRect, 98  
     XVT\_Dialog::SetInnerRect, 122  
     XVT\_DrawableContainer::SetInnerRect,  
         148

**N**

needs\_update()  
     XVT\_BaseDrawProto::NeedsUpdate, 30

**O**

open\_file\_dlg()  
     XVT\_GlobalAPI::OpenFile, 203

**P**

page\_setup\_dlg()  
     XVT\_GlobalAPI::PageSetup, 204  
picture\_close()  
     XVT\_Picture::Close, 302

picture\_lock()  
     XVT\_Picture::GetOpaqueData, 303  
     XVT\_Picture::GetOpaqueDataSize, 303  
     XVT\_Picture::Lock, 304  
picture\_unlock()  
     XVT\_Picture::GetOpaqueData, 303  
     XVT\_Picture::Unlock, 305  
process\_events()  
     XVT\_GlobalAPI::ProcessEvent, 205  
pt\_in\_rect()  
     XVT\_Rct::Contains, 325

**R**

rect\_intersect()  
     XVT\_Rct::Intersect, 329  
release\_mouse()  
     XVT\_ChildBase::ReleaseMouse, 55  
restore\_dir()  
     XVT\_GlobalAPI::RestoreDir, 207

**S**

save\_dir()  
     XVT\_GlobalAPI::SaveDir, 207  
save\_file\_dlg() XVT\_GlobalAPI::SaveFile, 208  
set\_caret\_dimensions()  
     XVT\_ChildBase::SetCaretDimensions, 56  
set\_clip()  
     XVT\_BaseDrawProto::SetClip, 32  
set\_cursor()  
     XVT\_ChildBase::SetCursor, 58  
set\_doc\_title()  
     XVT\_MenuWin::SetDocTitle, 289  
set\_event\_mask()  
     XVT\_Dialog::SetEventMask, 121  
     XVT\_DrawableContainer::SetEventMask,  
         147  
set\_front\_window()  
     XVT\_ChildBase::MakeFront, 55  
     XVT\_Control::MakeFront, 97  
set\_scroll\_pos()  
     XVT\_ChildBase::SetScrollPosition, 59  
     XVT\_ScrollBar::SetScrollPosition, 342

- set\_scroll\_proportion()
  - XVT\_ChildBase::SetScrollProportion, 60
  - XVT\_ScrollBar::SetScrollProportion, 343
- set\_scroll\_range()
  - XVT\_ChildBase::SetScrollRange, 61
  - XVT\_ScrollBar::SetScrollRange, 344
- set\_title()
  - XVT\_Dialog::SetTitle, 122
  - XVT\_Label::SetTitle, 241
  - XVT\_MenuWin::SetTitle, 291
- set\_value( ATTR\_PS\_PRINT\_FILE\_NAME )
  - XVT\_PrintWin::SetOutputFile, 315
- set\_value()
  - XVT\_GlobalAPI::SetAttrValue, 223
- show\_window()
  - XVT\_ChildBase::SetVisibleState, 62
  - XVT\_Control::SetVisibleState, 98
  - XVT\_Dialog::SetVisibleState, 123
- slist\_add()
  - XVT\_StrList::Add, 351
- slist\_add\_sorted()
  - XVT\_StrList::AddSorted, 352
- slist\_count()
  - XVT\_StrList::Count, 352
- slist\_dbg()
  - XVT\_StrList::Debug, 352
- slist\_elt()
  - XVT\_StrList::GetElement, 353
- slist\_first()
  - XVT\_StrList::GetFirst, 354
- slist\_next()
  - XVT\_StrList::GetNext, 355
- slist\_rem()
  - XVT\_StrList::Remove, 355
- start\_print\_thread()
  - XVT\_PrintWin::Init, 314
- startup\_dir()
  - XVT\_GlobalAPI::StartupDir, 224
- T**
- translate\_points()
  - XVT\_GlobalAPI::TranslatePoints, 225
- trap\_mouse()
  - XVT\_ChildBase::TrapMouse, 62
- tx\_add\_par()
  - XVT\_TextEdit::AddPar, 372
- tx\_append()
  - XVT\_TextEdit::Append, 372
- tx\_clear()
  - XVT\_TextEdit::Clear, 373
- tx\_create()
  - XVT\_TextEdit::Init, 383
- tx\_def\_create()
  - XVT\_TextEdit::Init, 383
- tx\_del\_par()
  - XVT\_TextEdit::DelPar, 374
- tx\_destroy()
  - XVT\_TextEdit::Close, 373
- tx\_get\_active()
  - XVT\_ChildBase::GetActiveTextEdit, 50
- tx\_get\_attrib() const
  - XVT\_TextEdit::GetAttrib, 375
- tx\_get\_border()
  - XVT\_TextEdit::GetBorder, 375
- tx\_get\_font()
  - XVT\_TextEdit::GetFont, 376
- tx\_get\_limit()
  - XVT\_TextEdit::GetLimit, 376
- tx\_get\_line()
  - XVT\_TextEdit::GetLine, 377
- tx\_get\_margin()
  - XVT\_TextEdit::GetMargin, 377
- tx\_get\_num\_chars()
  - XVT\_TextEdit::GetNumChars, 378
- tx\_get\_num\_lines()
  - XVT\_TextEdit::GetNumLines, 378
- tx\_get\_num\_par\_lines()
  - XVT\_TextEdit::GetNumParLines, 379
- tx\_get\_num\_pars()
  - XVT\_TextEdit::GetNumPars, 379
- tx\_get\_origin()
  - XVT\_TextEdit::GetOrigin, 380
- tx\_get\_sel()
  - XVT\_TextEdit::GetSel, 381



- tx\_get\_view()
  - XVT\_TextEdit::GetView, 381
- tx\_reset()
  - XVT\_TextEdit::Reset, 384
- tx\_resume()
  - XVT\_TextEdit::Resume, 384
- tx\_set\_active()
  - XVT\_TextEdit::SetActive, 385
- tx\_set\_attrib()
  - XVT\_TextEdit::SetAttrib, 385
- tx\_set\_border()
  - XVT\_TextEdit::SetBorder, 386
- tx\_set\_colors()
  - XVT\_TextEdit::SetColors, 386
- tx\_set\_font()
  - XVT\_TextEdit::SetFont, 387
- tx\_set\_limit()
  - XVT\_TextEdit::SetLimit, 387
- tx\_set\_margin()
  - XVT\_TextEdit::SetMargin, 388
- tx\_set\_par()
  - XVT\_TextEdit::SetPar, 388
- tx\_set\_sel()
  - XVT\_TextEdit::SetSel, 389
- tx\_suspend()
  - XVT\_TextEdit::Suspend, 389
- tx\_vscroll()
  - XVT\_TextEdit::DoVscroll, 375
- U**
- update\_window()
  - XVT\_BaseDrawProto::UpdateWindow, 36
- W**
- win\_move\_to()
  - XVT\_BaseDrawProto::SetCurrentPoint, 33
- win\_check\_box()
  - XVT\_CheckBox::SetCheckedState, 45
- win\_check\_radio\_button()
  - XVT\_RadioButton::SetCheckedState, 320
- win\_draw\_aline()
  - XVT\_BaseDrawProto::DrawALine, 16
- win\_draw\_arc()
  - XVT\_BaseDrawProto::DrawArc, 17
- win\_draw\_icon()
  - XVT\_BaseDrawProto::DrawIcon, 18
- win\_draw\_line()
  - XVT\_BaseDrawProto::DrawLine, 19
- win\_draw\_oval()
  - XVT\_BaseDrawProto::DrawOval, 20
- win\_draw\_pie()
  - XVT\_BaseDrawProto::DrawPie, 21
- win\_draw\_polygon()
  - XVT\_BaseDrawProto::DrawPolygon, 22
- win\_draw\_polyline()
  - XVT\_BaseDrawProto::DrawPolyLine, 23
- win\_draw\_rect()
  - XVT\_BaseDrawProto::DrawRect, 24
- win\_draw\_round\_rect()
  - XVT\_BaseDrawProto::DrawRoundedRect, 25
- win\_draw\_text()
  - XVT\_BaseDrawProto::DrawText, 26
- win\_get\_draw\_ctools()
  - XVT\_BaseDrawProto::GetBrush, 26
  - XVT\_BaseDrawProto::GetDrawMode, 27
  - XVT\_BaseDrawProto::GetDrawTools, 28
  - XVT\_BaseDrawProto::GetPen, 29
- win\_get\_font\_metrics()
  - XVT\_BaseDrawProto::GetFontMetrics, 28
- win\_get\_text\_width()
  - XVT\_BaseDrawProto::GetTextWidth, 29
- win\_list\_add()
  - XVT\_List::Add, 245
  - XVT\_ListEdit::Add, 264
- win\_list\_clear()
  - XVT\_List::Clear, 245
- win\_list\_count\_all()
  - XVT\_List::CountAll, 245
- win\_list\_count\_sel()
  - XVT\_List::CountSelections, 246
- win\_list\_delete()
  - XVT\_List::Delete, 246

win\_list\_get\_all()  
     XVT\_List::GetAll, 247  
 win\_list\_get\_elt()  
     XVT\_List::GetElement, 247  
 win\_list\_get\_first\_sel()  
     XVT\_List::GetFirstSelection, 248  
 win\_list\_get\_sel()  
     XVT\_List::GetSelections, 249  
 win\_list\_get\_sel\_index()  
     XVT\_List::GetSelectionIndex, 249  
 win\_list\_is\_sel()  
     XVT\_List::GetSelectedState, 249  
 win\_list\_resume()  
     XVT\_ListBox::SetSuspendedState, 255  
 win\_list\_set\_sel()  
     XVT\_List::SetSelectedState, 251  
 win\_list\_suspend()  
     XVT\_ListBox::SetSuspendedState, 255  
 win\_menu\_check()  
     XVT\_MenuItem::SetCheckedState, 275  
 win\_menu\_enable()  
     XVT\_MenuNode::SetEnabledState, 279  
 win\_menu\_fetch()  
     XVT\_MenuWin::GetMenu, 288  
 win\_menu\_show()  
     XVT\_MenuWin::SetMenu, 290  
 win\_picture\_draw()  
     XVT\_BaseDrawProto::DrawPicture, 21  
 win\_scroll\_rect()  
     XVT\_DrawableContainer::Scroll, 146  
 win\_select\_item\_text()  
     XVT\_Editable::SelectText, 163  
 win\_set\_back\_color()  
     XVT\_BaseDrawProto::SetBackColor, 31  
 win\_set\_cbrush()  
     XVT\_BaseDrawProto::SetBrush, 31  
 win\_set\_cpen()  
     XVT\_BaseDrawProto::SetPen, 36  
 win\_set\_draw\_ctools()  
     XVT\_BaseDrawProto::SetDrawTools, 34  
 win\_set\_draw\_mode()  
     XVT\_BaseDrawProto::SetDrawMode, 34

win\_set\_font()  
     XVT\_BaseDrawProto::SetFont, 35  
 win\_set\_font\_menu()  
     XVT\_MenuWin::SetFontMenu, 290  
 win\_set\_fore\_color()  
     XVT\_BaseDrawProto::SetForeColor, 35  
 win\_set\_menu\_text()  
     XVT\_MenuNode::SetTitle, 280

**X**

xvt\_ask()  
     XVT\_GlobalAPI::Ask, 184  
 xvt\_beep()  
     XVT\_GlobalAPI::Beep, 185  
 xvt\_dbg()  
     XVT\_GlobalAPI::Debug, 186  
 xvt\_error()  
     XVT\_GlobalAPI::Error, 187  
 xvt\_fatal()  
     XVT\_GlobalAPI::Fatal, 187  
 xvt\_help()  
     XVT\_GlobalAPI::Help, 200  
 xvt\_msg()  
     XVT\_GlobalAPI::Message, 202  
 xvt\_note()  
     XVT\_GlobalAPI::Note, 202  
 xvt\_system()  
     XVT\_TaskWin::Init, 366  
 xvt\_terminate()  
     XVT\_TaskWin::Close, 363



# XVT++

## GENERAL INDEX

.frl file, 88

.hlp file, 88

.uid file, 88

\_ScreenWin

from XVT\_Base, 11, 333

\_TaskWin

from XVT\_Base, 12

### A

About

from XVT\_GlobalAPI, 183

about box dialog

displaying, 183

retrieving resource ID, 85

setting resource ID, 87

abstract class, vi

accelerator key, 278

Activity

from XVT\_TextEdit, 371

Add

from XVT\_List, 244

from XVT\_ListEdit, 263

from XVT\_StrList, 350

add

from StrList, 510

adding

item to list, 244, 263, 510

item to list box, 442

item to string list, 350

item to string list in order, 351, 511

paragraph to text edit object, 371

string to paragraph, 372

AddPar

from XVT\_TextEdit, 371

AddSorted

from XVT\_StrList, 351

alert box, displaying, 186, 187, 202

allocating global memory, 189

AnotherPage

from XVT\_PrintWin, 311

Append

from XVT\_TextEdit, 372

appending

debug information, 185

debug information, conditionally, 186

string list to debug file, 352

application

closing, 362

error, 187

notifying object of creation, 363

notifying object of exit, 364

retrieving base name, 86

retrieving name, 85

setting base name, 88

setting name, 88

starting, 520

- arc
  - drawing, 17, 469
  - from GraphWin, 469
- area, invalidating, 144
- ascent
  - retrieving, 28, 179
  - setting, 180
- ASCII characters, 26, 75, 110, 130
- Ask
  - from XVT\_GlobalAPI, 183
- aspect ratio, 20
- ATTR\_\* constants, 209
  - ATTR\_BACK\_COLOR, 210
  - ATTR\_CH\_\*, 209
  - ATTR\_CTL\_BUTTON\_HEIGHT, 211
  - ATTR\_CTL\_CHECK\_BOX\_HEIGHT, 211
  - ATTR\_CTL\_EDIT\_TEXT\_HEIGHT, 211
  - ATTR\_CTL\_HORZ\_SBAR\_HEIGHT, 212
  - ATTR\_CTL\_RADIOBUTTON\_HEIGHT, 212
  - ATTR\_CTL\_STATIC\_TEXT\_HEIGHT, 212
  - ATTR\_CTL\_VERT\_SBAR\_WIDTH, 213
  - ATTR\_DBLFRAME\_HEIGHT, 213
  - ATTR\_DBLFRAME\_WIDTH, 213
  - ATTR\_DEBUG\_FILENAME, 213
  - ATTR\_DOC\_STAGGER\_HORZ, 214
  - ATTR\_DOC\_STAGGER\_VERT, 215
  - ATTR\_DOCFRAME\_HEIGHT, 214
  - ATTR\_DOCFRAME\_WIDTH, 214
  - ATTR\_EVENT\_HOOK, 215
  - ATTR\_FATAL\_ERR\_HANDLER, 215
  - ATTR\_FRAME\_HEIGHT, 216
  - ATTR\_FRAME\_WIDTH, 216
  - ATTR\_HAVE\_COLOR, 216
  - ATTR\_HAVE\_MOUSE, 217
  - ATTR\_ICON\_HEIGHT, 217
  - ATTR\_ICON\_WIDTH, 217
  - ATTR\_KEY\_HOOK, 218
  - ATTR\_MAC\_\*, 209
  - ATTR\_MALLOC\_ERR\_HANDLER, 218
  - ATTR\_MENU\_HEIGHT, 219
  - ATTR\_NATIVE\_GRAPHIC\_CONTEXT, 219
  - ATTR\_NATIVE\_WINDOW, 220
  - ATTR\_NUM\_TIMERS, 220
  - ATTR\_PM\_\*, 209
  - ATTR\_PRINTER\_HEIGHT, 220
  - ATTR\_PRINTER\_HRES, 221
  - ATTR\_PRINTER\_VRES, 221
  - ATTR\_PRINTER\_WIDTH, 221
  - ATTR\_SCREEN\_HEIGHT, 222
  - ATTR\_SCREEN\_HRES, 222
  - ATTR\_SCREEN\_VRES, 222
  - ATTR\_SCREEN\_WIDTH, 222
  - ATTR\_SUPPRESS\_UPDATE\_CHECK, 223
  - ATTR\_TITLE\_HEIGHT, 223
  - ATTR\_WIN\_\*, 209
  - ATTR\_WIN\_PM\_\*, 210
  - ATTR\_X\_\*, 210
  - ATTR\_XM\_\*, 210
  - ATTR\_XOL\_\*, 210
- ATTR\_KEY\_HOOK, 110, 131
- ATTR\_MALLOC\_ERR\_HANDLER, 9
- attribute
  - retrieving, 190
  - setting, 208
- B**
  - background color
    - retrieving, 151, 191
    - setting, 30, 153
  - BaseWin class, 3, 414
  - BaseXVT class, 424
  - Beep
    - from XVT\_GlobalAPI, 184
  - begin
    - from TaskWin, 520
  - behavior objects, 2
  - blue
    - retrieving color component, 81
    - setting color component, 82
  - bottom
    - from Rct, 494

boundary, retrieving for client area, 12

breaking array into lines, 188

brush

converting, 433

from Brush, 433

from DrawTools, 460

retrieving, 26, 151, 460

retrieving color, 38

retrieving pattern, 38

setting, 31, 154, 460, 475

setting color, 39

setting pattern, 39

Brush class, 432

button, notifying object of, 42

## C

caret

retrieving position, 50

retrieving visibility, 51

setting dimensions, 56

setting position, 56

cast, vii, 10

CB\_\* formats, 75

CBRUSH structure, 433

CCHELP, 199

changing directory, 185

character input, notifying object, 109, 130

check

from Control, 437

from Font, 464

check box

checking, 45

determining if checked, 441

unchecking, 45

checkbox

checking, 437

unchecking, 451

checking

check box, 45

checkbox, 437

menu item, 275, 483

radio button, 437

ChgDir

from XVT\_GlobalAPI, 185

child window

initializing, 67

retrieving, 334

retrieving first of, 142

retrieving next, 143, 334

retrieving number, 143, 335

class heirarchy, 3

Clear, 154, 210

from XVT\_BaseDrawProto, 129

from XVT\_List, 245

from XVT\_TextEdit, 373

clear

from GraphWin, 470

clearing a window, 129, 470

client area

retrieving, 418

retrieving boundary, 12

clipboard

closing, 74

determining if format available, 74

determining if open, 77

opening, 77

putting data on, 78

retrieving data, 76

clipping, 32

retrieving state, 27

setting state, 32

clipping region

retrieving, 26

setting, 31

Close, 111, 286

from XVT\_ClipBoard, 74

from XVT\_Control, 92

from XVT\_Dialog, 109

from XVT\_DrawableContainer, 129

from XVT\_Picture, 302

from XVT\_TaskWin, 362

from XVT\_TextEdit, 373

- close
  - from BaseXVT, 425
  - from Control, 438
  - notifying object of request, 111, 286
- closing
  - application, 362
  - clipboard, 74
  - notifying object, 363
  - object, 425
- code, typesetting convention, v
- color
  - from Brush, 434
  - from Pen, 487
  - retrieving background, 151, 191
  - retrieving blue component, 81
  - retrieving for brush, 38, 434
  - retrieving for pen, 296, 487
  - retrieving foreground, 152
  - retrieving green component, 81
  - retrieving red component, 82
  - setting background, 30, 153
  - setting blue component, 82
  - setting for brush, 39, 434
  - setting for pen, 297, 487
  - setting foreground, 35, 155
  - setting green component, 82
  - setting red component, 83
- compatibility
  - 1.1 classes, 407
  - discussion of, 3
- concrete class, vi
- configuration data, retrieving, 520
- Constrain
  - from XVT\_Rct, 62, 324
- constraining point to a rectangle, 324
- constructor, vii
- container
  - notifying object of creation, 131
  - notifying object of destruction, 132
  - retrieving event mask, 141
  - setting event mask, 146
- Contains
  - from XVT\_Rct, 325
- control
  - creating, 438, 439
  - destroying, 92, 438
  - determining visibility, 96
  - determining whether enabled, 95
  - disabling, 97
  - enabling, 97
  - initializing, 96
  - notifying object of, 417
  - notifying object of creation, 93
  - notifying object of destruction, 93
  - notifying object of user-defined event, 94
  - retrieving, 115
  - retrieving first, 141
  - retrieving first in a dialog, 117
  - retrieving next, 142
  - retrieving number, 116, 140
  - retrieving subsequent in dialog, 117
  - retrieving title, 239
  - retrieving user data, 196
  - setting focus, 96
  - setting title, 240
  - setting visibility, 98
- Control class, 436
- conventions, typesetting, v
- converting
  - brush, 433
  - picture to opaque data, 302
- coordinates, setting, 491
- Count
  - from XVT\_StrList, 352
- count
  - from StrList, 511
- CountAll
  - from XVT\_List, 245
- counting
  - controls in dialog, 116
  - elements in string list, 352
  - items in list box, 443
  - selected items in list box, 444

CountSelections  
    from XVT\_List, 246

create  
    from DlgWin, 454  
    from ScreenWin, 500

create\_def  
    from Control, 438  
    from DlgWin, 455  
    from ScreenWin, 501

create\_scratch  
    from Control, 439  
    from ScreenWin, 502

creating  
    control, 438, 439  
    dialog, 454, 455  
    GUI object, 2  
    window, 500, 501, 502

cursor, setting shape, 57  
CURSOR\_\* constants, 58

## D

data, clipboard, 76, 78

dbg  
    from StrList, 512

deactivation, notifying object of, 418

Debug  
    from XVT\_GlobalAPI, 185  
    from XVT\_StrList, 352

Debug2  
    from XVT\_GlobalAPI, 186

default directory, retrieving, 192

delegate objects, 2

Delete  
    from XVT\_List, 246

deleting, paragraph, 373

DelPar  
    from XVT\_TextEdit, 373

descent  
    retrieving, 28, 179  
    setting, 180

deselecting an item, 250

destroying  
    control, 92, 438  
    dialog, 109  
    window, 129

destructor, vii

dialog  
    about box, displaying, 183  
    creating, 454, 455  
    destroying, 109  
    determining visibility, 118  
    determining whether enabled, 116  
    disabling, 120  
    enabling, 120  
    initializing, 119  
    notifying object of creation, 111  
    notifying object of destruction, 112  
    page setup, displaying, 204  
    retrieving first control in, 117  
    retrieving number of controls, 116  
    retrieving subsequent controls, 117  
    retrieving title, 118  
    retrieving user data, 193  
    setting position, 121  
    setting size, 121  
    setting title, 122  
    setting visibility, 123

Difference  
    from XVT\_Rct, 326

dimensions  
    retrieving for rectangle, 327  
    setting for rectangle, 495

directory  
    changing, 185  
    restoring, 206  
    retrieving, 172, 194  
    retrieving default, 192  
    saving, 207  
    setting, 173  
    startup, returning to, 224

DirToStr  
    from XVT\_Directory, 126



- disable
  - from BaseXVT, 425
  - from Menu Item, 483
- disabling
  - control, 97
  - dialog, 120
  - menu item, 279, 483, 484
  - object, 425
  - window, 58
- dispatch
  - from BaseWin, 408, 415
- dispatching an event, 415
- displaying
  - about box dialog, 183
  - alert box and terminating, 187
  - alert box with error icon, 186
  - alert box with note icon, 202
  - emergency message, 201
  - page setup dialog, 204
- DLG\_CANCEL control ID, 183
- DLG\_OK control ID, 183
- DlgWin class, 3, 453
- DoDraw function, 128
- DoHScroll
  - from XVT\_TextEdit, 374
- downcasting, 10
- DRAW\_MODE enumeration, 33, 156
- DrawAction, 128
  - from XVT\_PrintWin, 311
- DrawALine
  - from XVT\_BaseDrawProto, 16
- DrawArc
  - from XVT\_BaseDrawProto, 17
- DrawIcon, 217
  - from XVT\_BaseDrawProto, 18
- drawing
  - arc, 17, 469
  - icon, 18, 470
  - line, 16, 19, 471
  - oval, 19, 472
  - picture, 20
  - pie, 21, 473
  - polygon, 22, 473
  - polyline, 23, 474
  - rectangle, 23, 474
  - rounded rectangle, 24, 475
  - stop recording primitives, 302
  - text string, 25, 478
  - window, 128
- drawing mode
  - retrieving, 27, 152, 461
  - setting, 33, 155, 461, 476
- drawing tools
  - retrieving, 28, 470
  - setting, 34, 477
- DrawInit
  - from XVT\_PrintWin, 312
- DrawLine
  - from XVT\_BaseDrawProto, 19
- DrawOval
  - from XVT\_BaseDrawProto, 19
- DrawPicture
  - from XVT\_BaseDrawProto, 20
- DrawPie
  - from XVT\_BaseDrawProto, 21
- DrawPolygon
  - from XVT\_BaseDrawProto, 22
- DrawPolyline
  - from XVT\_BaseDrawProto, 23
- DrawProtocol
  - from XVT\_DrawableContainer, 128
  - from XVT\_PrintWin, 310
- DrawRect
  - from XVT\_BaseDrawProto, 23
- DrawRoundedRect
  - from XVT\_BaseDrawProto, 24
- DrawText
  - from XVT\_BaseDrawProto, 25
- DrawTools class, 459

**E**

- e\_action, 2, 253
  - from XVT\_Button, 42
  - from XVT\_ListBox, 254
  - from XVT\_ListButton, 259
  - from XVT\_MenuItem, 274
  - from XVT\_ScrollBar, 49, 50, 338
  - from XVT\_Toggle, 394
- e\_activate
  - from BaseWin, 416
- e\_char, 59, 62, 120, 123
  - from XVT\_Dialog, 109
  - from XVT\_DrawableContainer, 130
- e\_close, 129
  - from XVT\_Dialog, 111
  - from XVT\_TaskWin, 363
- e\_command
  - from BaseWin, 416
- e\_control
  - from BaseWin, 417
- e\_create, 1, 67, 68, 103, 114, 119, 136, 286, 362, 398, 402
  - from XVT\_Control, 93
  - from XVT\_Dialog, 111
  - from XVT\_DrawableContainer, 131
  - from XVT\_TaskWin, 363
- e\_deactivate
  - from BaseWin, 418
- e\_destroy, 1, 93, 109, 111, 130, 363, 399
  - from XVT\_Control, 93
  - from XVT\_Dialog, 112
  - from XVT\_DrawableContainer, 132
  - from XVT\_TaskWin, 364
- e\_focus, 59, 111, 120, 123
  - from XVT\_Dialog, 112
  - from XVT\_DrawableContainer, 132
  - from XVT\_Editable, 162
- e\_font, 154, 175
  - from XVT\_MenuWin, 287
- e\_hscroll, 146
  - from XVT\_MenuWin, 49
- e\_mouse\*, 59
  - e\_mouse\_\*, 62
  - e\_mouse\_dbl
    - from XVT\_DrawableContainer, 133
  - e\_mouse\_down
    - from XVT\_DrawableContainer, 134
  - e\_mouse\_move, 62
    - from XVT\_DrawableContainer, 135
  - e\_mouse\_up
    - from XVT\_DrawableContainer, 135
  - e\_quit, 367
    - from XVT\_TaskWin, 364
  - e\_size
    - from XVT\_Dialog, 113
    - from XVT\_DrawableContainer, 136
  - e\_timer, 390
    - from XVT\_Dialog, 114
    - from XVT\_DrawableContainer, 137
  - e\_update, 30, 36, 128, 145, 154, 223
    - from XVT\_DrawableContainer, 137
  - e\_user
    - from XVT\_Control, 94
    - from XVT\_Dialog, 115
    - from XVT\_DrawableContainer, 139
  - e\_vscroll
    - from XVT\_MenuWin, 49
- elt
  - from StrList, 512
- EM\_\* constants, 121, 146
- emergency message, 201
- emphasis, typesetting convention, v
- empty
  - from Rct, 494
- enable
  - from BaseXVT, 425
  - from Menu item, 484
- enabling
  - control, 97
  - dialog, 120
  - menu item, 279, 484
  - object, 425
  - window, 58
- entering help system, 199

- EOL, finding, 188
- EOL\_\* values, 188
- EOL\_SEQ constant, 75
- Error
  - from XVT\_GlobalAPI, 186
- error
  - application, 187
  - handling, 6, 167, 234
  - raising, 9, 170, 237
- event
  - dispatching, 415
  - notifying object of, 115, 139
  - processing, 204
  - user-defined, 94, 115
- event delivery mask
  - retrieving, 419
  - setting, 420
- event handlers, 2
- event hook, 215
- event mask
  - constants, 146
  - retrieving, 116, 141
  - setting, 120, 146
- event mask constants, 121
- F**
- family
  - from Font, 464
- Fatal, 187
  - from XVT\_GlobalAPI, 187
- Fatal function, 215
- file
  - determining if readable, 205
  - opening, 203
  - saving, 207
  - setting type, 224
- file name
  - retrieving, 172
  - setting, 174
  - typesetting convention, v
- file type
  - retrieving, 173
  - setting, 174
- FindEOL
  - from XVT\_GlobalAPI, 188
- FindEOL function, 75
- finding EOL, 188
- First
  - from XVT\_StrList, 353
- first
  - from StrList, 513
- focus, notifying object of change, 112, 132
- font
  - from DrawTools, 460
  - notifying object of change, 287
  - retrieving, 152, 460
  - retrieving size, 176
  - selecting, 465
  - setting, 34, 154, 419, 460, 476
  - setting size, 176
- Font class, 463
- font family
  - retrieving, 464
  - setting, 464
- font metrics, retrieving, 28, 503
- font size
  - retrieving, 466
  - setting, 466
- font structure, retrieving, 465
- font style
  - retrieving, 466
  - setting, 466
- foreground color
  - retrieving, 152
  - setting, 35, 155
- format, clipboard, 74
- freeing global memory, 197
- G**
- GAlloc
  - from XVT\_GlobalAPI, 189
- GC, 219
- get
  - from StrList, 408, 513
- get\_client
  - from BaseWin, 418

- get\_config
  - from TaskWin, 520
- get\_def
  - from BaseXVT, 426
- get\_font
  - from Font, 465
- get\_mask
  - from BaseWin, 419
- get\_metrics
  - from ScreenWin, 503
- get\_rect
  - from BaseXVT, 426
- get\_scroll\_pos
  - from Control, 440
- get\_scroll\_proportion
  - from Control, 440
- get\_scroll\_range
  - from Control, 441
- get\_text
  - from BaseXVT, 427
- get\_tools
  - from GraphWin, 470
- get\_type
  - from BaseXVT, 408, 427
- get\_win
  - from BaseWin, 419
- GetAboutBoxID
  - from XVT\_Config, 85
- GetActiveTextEdit
  - from XVT\_ChildBase, 50
- GetAll
  - from XVT\_List, 247
- GetApplName
  - from XVT\_Config, 85
- GetAscent
  - from XVT\_FontMetrics, 179
- GetAttrib
  - from XVT\_TextEdit, 375
- GetAttrValue
  - from XVT\_GlobalAPI, 190
- GetBackColor
  - from XVT\_DrawTools, 151
- GetBaseApplName
  - from XVT\_Config, 86
- GetBlue
  - from XVT\_Color, 81
- GetBorder
  - from XVT\_TextEdit, 375
- GetBottomLeft
  - from XVT\_Rct, 326
- GetBottomRight
  - from XVT\_Rct, 327
- GetBrush
  - from XVT\_BaseDrawProto, 26
  - from XVT\_DrawTools, 151
- GetCaretPos
  - from XVT\_ChildBase, 50
- GetCaretState
  - from XVT\_ChildBase, 51
- GetCheckableState
  - from XVT\_MenuItem, 275
- GetCheckedState
  - from XVT\_MenuItem, 275
  - from XVT\_Toggle, 394
- GetClip
  - from XVT\_BaseDrawProto, 26
- GetClipState
  - from XVT\_BaseDrawProto, 27
- GetColor
  - from XVT\_Brush, 38
  - from XVT\_Pen, 296
- GetCount
  - from XVT\_Menu, 270
- GetCtl
  - from XVT\_Dialog, 115
  - from XVT\_DrawableContainer, 140
- GetCtlCount
  - from XVT\_Dialog, 116
  - from XVT\_DrawableContainer, 140
- GetCurrentPoint
  - from XVT\_BaseDrawProto, 27
- GetData
  - from XVT\_ClipBoard, 76

- GetDefaultBackColor
  - from XVT\_GlobalAPI, 191
- GetDefaultDir
  - from XVT\_GlobalAPI, 192
- GetDescent
  - from XVT\_FontMetrics, 179
- GetDialogUserData
  - from XVT\_GlobalAPI, 193
- GetDimVect
  - from XVT\_Rct, 327
- GetDir, 192
  - from XVT\_FileSpec, 172
  - from XVT\_GlobalAPI, 194
- GetDrawMode
  - from XVT\_BaseDrawProto, 27
- GetDrawTools, 154, 175
  - from XVT\_BaseDrawProto, 28
- GetElement
  - from XVT\_List, 247
  - from XVT\_StrList, 353
- GetEnabledState
  - from XVT\_ChildBase, 51
  - from XVT\_Control, 95
  - from XVT\_Dialog, 116
  - from XVT\_MenuNode, 278
- GetEventMask
  - from XVT\_Dialog, 116
  - from XVT\_DrawableContainer, 141
- GetFirst
  - from XVT\_Menu, 270
- GetFirstCtl
  - from XVT\_Dialog, 117
  - from XVT\_DrawableContainer, 141
- GetFirstSelection
  - from XVT\_List, 248
- GetFirstWin
  - from XVT\_DrawableContainer, 142
  - from XVT\_ScreenWin, 334
- GetFont, 154
  - from XVT\_DrawTools, 152
  - from XVT\_TextEdit, 376
- GetFontMetrics
  - from XVT\_BaseDrawProto, 28, 178
- GetForeColor
  - from XVT\_DrawTools, 152
- GetFrontTopLevelWin
  - from XVT\_GlobalAPI, 191
- GetFrontWin
  - from XVT\_GlobalAPI, 192
- GetGreen
  - from XVT\_Color, 81
- GetID
  - from XVT\_Control, 95
- GetInnerRect
  - from XVT\_Base, 12
- GetInterval
  - from XVT\_Timer, 392
- GetItem
  - from XVT\_Menu, 270
- GetLeading
  - from XVT\_FontMetrics, 180
- GetLimit
  - from XVT\_TextEdit, 376
- GetLine
  - from XVT\_TextEdit, 376
- GetLockedState
  - from XVT\_Picture, 302
- GetMargin
  - from XVT\_TextEdit, 377
- GetMenu
  - from XVT\_MenuWin, 287
- GetMenuBarID
  - from XVT\_Config, 86
- GetMenuUserData
  - from XVT\_GlobalAPI, 194
- GetMKey
  - from XVT\_MenuNode, 278
- GetMode
  - from XVT\_DrawTools, 152
- GetName
  - from XVT\_FileSpec, 172
- GetNext
  - from XVT\_Menu, 271

- GetNextCtl
  - from XVT\_Dialog, 117
  - from XVT\_DrawableContainer, 142
- GetNextWin
  - from XVT\_DrawableContainer, 143
  - from XVT\_ScreenWin, 334
- GetNumChars
  - from XVT\_TextEdit, 378
- GetNumLines
  - from XVT\_TextEdit, 378
- GetNumParLines
  - from XVT\_TextEdit, 379
- GetNumPars
  - from XVT\_TextEdit, 379
- GetOpaqueData
  - from XVT\_Picture, 302
- GetOpaqueDataSize, 303
  - from XVT\_Picture, 303
- GetOpaqueText
  - from XVT\_DrawTools, 153
- GetOpenState
  - from XVT\_ClipBoard, 77
  - from XVT\_Picture, 304
- GetOrigin
  - from XVT\_TextEdit, 380
- GetOuterRect
  - from XVT\_Base, 13
- GetOutputFile
  - from XVT\_PrintWin, 312
- GetParent
  - from XVT\_ChildBase, 51
  - from XVT\_Control, 95
  - from XVT\_MenuNodeBase, 282
- GetPattern
  - from XVT\_Brush, 38
  - from XVT\_Pen, 296
- GetPen
  - from XVT\_BaseDrawProto, 28
  - from XVT\_DrawTools, 153
- GetPrintRcd
  - from XVT\_PrintWin, 313
- GetPrintRcdSize
  - from XVT\_PrintWin, 313
- GetRed
  - from XVT\_Color, 82
- GetResString
  - from XVT\_GlobalAPI, 195
- GetScrollPosition
  - from XVT\_ScrollBar, 339
  - from XVT\_ChildBase, 52
- GetScrollProportion
  - from XVT\_ChildBase, 52
  - from XVT\_ScrollBar, 340
- GetScrollRange
  - from XVT\_ChildBase, 53
  - from XVT\_ScrollBar, 340
- GetSel
  - from XVT\_TextEdit, 380
- GetSelectedState
  - from XVT\_List, 248
- GetSelectionIndex
  - from XVT\_List, 249
- GetSelections
  - from XVT\_List, 249
- GetSize
  - from XVT\_Font, 176
- GetStyle
  - from XVT\_Pen, 296
- GetSubMenuPtr
  - from XVT\_SubMenu, 357
- GetSuspendedState
  - from XVT\_ListBox, 254
- GetTaskWinTitle
  - from XVT\_Config, 87
- GetTextEdit
  - from XVT\_ChildBase, 54
- GetTextWidth, 211, 212
  - from XVT\_BaseDrawProto, 29
- GetTitle
  - from XVT\_Dialog, 118
  - from XVT\_Label, 239
  - from XVT\_MenuNode, 278
  - from XVT\_MenuWin, 288

- GetTopLeft
  - from XVT\_Rct, 327
- GetTopRight
  - from XVT\_Rct, 328
- GetType
  - from XVT\_Base, 13
  - from XVT\_FileSpec, 173
- GetView
  - from XVT\_TextEdit, 381
- GetVisibleState
  - from XVT\_ChildBase, 54
  - from XVT\_Control, 96
  - from XVT\_Dialog, 118
- GetWidth
  - from XVT\_Pen, 297
- GetWinCount
  - from XVT\_DrawableContainer, 143
  - from XVT\_ScreenWin, 335
- GetWindowUserData
  - from XVT\_GlobalAPI, 196
- GetX
  - from XVT\_Pnt, 307
- GetY
  - from XVT\_Pnt, 307
- GFree
  - from XVT\_GlobalAPI, 197
- GHANDLE, 189
- global memory
  - allocating, 189
  - freeing, 197
  - locking, 197
  - resizing, 198
  - retrieving size of block, 198
  - unlocking, 199
- GLock
  - from XVT\_GlobalAPI, 197
- Grafport, 219
- GraphWin class, 468
- GReAlloc
  - from XVT\_GlobalAPI, 198
- green
  - retrieving color component, 81
  - setting color component, 82
- GSize
  - from XVT\_GlobalAPI, 198
- gstrcat, 75
- GUnlock
  - from XVT\_GlobalAPI, 199
- H**
- Handler, 233
  - from XVT\_AllocErrorHandler, 6
  - from XVT\_ErrorHandle, 167
  - from XVT\_InternalErrorHandler, 234
- handlers, 2
- handling
  - error, 167
  - internal error, 234
  - memory allocation error, 6
- HDC, 219
- header file, vi
- Height
  - from XVT\_Rct, 328
- heirarchy of classes, 3
- Help
  - from XVT\_GlobalAPI, 199
- hide
  - from BaseXVT, 428
- hiding an object, 428, 430
- horizontal scrollbar, notifying object of activity, 49
- HPS, 219
- HWND, 220
- I**
- icon
  - drawing, 18, 470
  - from GraphWin, 470
  - initializing, 230
- ID
  - control's, retrieving, 95, 441
  - resource's, retrieving, 85, 86
  - resource's, setting, 87, 89

- id
  - from Control, 441
- IDT, 1
- if statement, 2
- implementation class, vi
- implementation members, viii
- inherited member function, viii
- Init
  - from XVT\_ChildWin, 67
  - from XVT\_Control, 96, 230, 240, 250, 341
  - from XVT\_DetachedWin, 102
  - from XVT\_Dialog, 119
  - from XVT\_Icon, 230
  - from XVT\_Label, 239
  - from XVT\_PrintWin, 314
  - from XVT\_ScrollBar, 341
  - from XVT\_TaskWin, 366
  - from XVT\_TextEdit, 382
  - from XVT\_TopLevelWin, 401
- initializing
  - child window, 67
  - control, 96
  - dialog, 119
  - icon, 230
  - label, 239
  - print window, 314
  - scrollbar, 341
  - task window, 366
  - text edit object, 382
  - top-level window, 401
  - window, 102
- inner rectangle
  - retrieving, 12
  - setting for control, 98
  - setting for dialog, 121
- Install
  - from XVT\_Menu, 271
- installing
  - item in menu, 271
- Interactive Design Tool, 1
- Intersect
  - from XVT\_Rct, 328
- Invalidate, 138
  - from XVT\_BaseDrawProto, 144
- invalidating
  - area, 144
  - disjoint areas of the window, 139
- invisibility
  - determining for control, 96
  - determining for dialog, 118
  - determining for window, 54
  - retrieving for caret, 51
  - setting for caret, 57
  - setting for dialog, 123
  - setting for window, 61
- is\_checked
  - from Control, 441
- IsEmpty
  - from XVT\_Rct, 329
- K**
- key, accelerator, 278
- L**
- label, initializing, 239
- lbox\_add
  - from Control, 442
- lbox\_clear
  - from Control, 443
- lbox\_count\_all
  - from Control, 443
- lbox\_count\_sel
  - from Control, 444
- lbox\_delete
  - from Control, 444
- lbox\_get\_all
  - from Control, 444
- lbox\_get\_elt
  - from Control, 445
- lbox\_get\_first\_sel
  - from Control, 445
- lbox\_get\_sel
  - from Control, 446
- lbox\_get\_sel\_index
  - from Control, 446



- lbox\_is\_sel
  - from Control, 447
- lbox\_resume
  - from Control, 447
- lbox\_set\_sel
  - from Control, 448
- lbox\_suspend
  - from Control, 448
- leading
  - retrieving, 28, 180
  - setting, 181
- left
  - from Rct, 494
- line
  - drawing, 16, 19, 471
  - from GraphWin, 471
- line-end sequence, 188
- list
  - adding elements, 510
  - adding items, 244, 263
  - removing item, 246
  - removing items, 245
  - retrieving all items, 247
  - retrieving item, 247
  - retrieving number of elements, 511
  - retrieving number of items, 245
- list box
  - adding items, 442
  - determining if updates are suspended, 254
  - notifying object of activity, 254
  - removing all items, 443, 444
  - removing item, 444
  - resuming updates, 255, 447
  - retrieving all selected items, 446
  - retrieving first selected item, 248, 445
  - retrieving index of first selected item, 446
  - retrieving number of items, 443
  - retrieving number of selected items, 444
  - suspending updates, 255, 448
- list button, notifying object of activity, 259
- ListFaces
  - from XVT\_GlobalAPI, 200

- listing typefaces, 200
- ListResStrings
  - from XVT\_GlobalAPI, 201
- Lock
  - from XVT\_Picture, 304
- locking
  - global memory, 197
  - picture data, 304
- longjmp, 167

## M

- M\_\* (drawing modes), 33, 156
- MakeFront, 113, 133
  - from XVT\_ChildBase, 55
  - from XVT\_Control, 96
- member function, vii
  - description format, viii
  - inherited, viii
- memory
  - allocating, 189
  - freeing, 197
  - locking, 197
  - resizing, 198
  - retrieving size of global block, 198
  - unlocking, 199
- memory allocation errors, 6
- menu
  - checking item, 275
  - determining if enabled, 278
  - determining if item is checkable, 275
  - determining if item is checked, 275
  - disabling item, 279
  - enabling item, 279
  - installing item, 271
  - notifying object of selection, 274, 416
  - replacing default item, 272
  - retrieving, 287
  - retrieving accelerator key, 278
  - retrieving first item, 270
  - retrieving item, 270
  - retrieving number of items, 270
  - retrieving parent, 282
  - retrieving subsequent items, 271

- setting, 290
  - unchecking item, 275
- menu item
  - checking, 483
  - disabling, 483, 484
  - enabling, 484
  - retrieving tag, 484
  - retrieving user data, 194
  - unchecking, 483, 484
- menubar
  - retrieving resource ID, 86
  - setting resource ID, 89
- MenuItem class, 482
- Message, 187
  - from XVT\_GlobalAPI, 201
- message, emergency, 201
- metrics, font, 503
- mode
  - drawing, 27, 33, 152, 155, 461, 476
  - from DrawTools, 461
- mouse
  - notifying object of double click, 133
  - notifying object of down event, 134
  - notifying object of move, 135
  - notifying object of up event, 135
  - releasing, 55
  - trapping, 62
- move
  - from BaseXVT, 428
- move\_to
  - from GraphWin, 472
- moving
  - current position, 472
  - object, 428
- MyButton, 391
- N**
- NeedsUpdate
  - from XVT\_BaseDrawProto, 30
- Next
  - from XVT\_StrList, 354
- next
  - from StrList, 514

- Normalize
  - from XVT\_Rct, 12, 329
- Note
  - from XVT\_GlobalAPI, 202
- O**
- object
  - closing, 425
  - disabling, 425
  - enabling, 425
  - hiding, 428, 430
  - moving, 428
  - retrieving, 140
  - retrieving title, 427
  - setting title, 429
  - showing, 430
- offset, retrieving for view rectangle, 380
- opaque data, determining buffer size, 303
- opaque text flag
  - retrieving, 153
  - setting, 156
- Open
  - from XVT\_ClipBoard, 77
- OpenFile, 200
  - from XVT\_GlobalAPI, 203
- opening
  - clipboard, 77
  - file, 203
- operator, vii
- outer rectangle, retrieving, 13, 426
- output file
  - retrieving, 312
  - retrieving name, 311
  - setting name, 314
- oval
  - drawing, 19, 472
  - from GraphWin, 472
- overloaded operator, vii
- P**
- page, printing another, 311
- PageSetup, 309
  - from XVT\_GlobalAPI, 204

- parent
  - from BaseXVT, 428
- parent window, retrieving, 51, 95, 428
- pat
  - from Brush, 434
  - from Pen, 487
- PAT\_STYLE enumeration, 39, 298
- pattern
  - retrieving for brush, 38, 434
  - retrieving for pen, 296, 487
  - setting for brush, 39, 434
  - setting for pen, 297, 487
- pen
  - from DrawTools, 461
  - retrieving, 28, 153, 461
  - retrieving color, 296, 487
  - retrieving pattern, 296, 487
  - retrieving position, 27
  - retrieving style, 296
  - retrieving width, 297, 488
  - setting, 36, 157, 461, 477
  - setting color, 297, 487
  - setting pattern, 297, 487
  - setting position, 32
  - setting style, 298
  - setting width, 299, 488
- Pen class, 486
- pen position, 32
- picture
  - converting to opaque data, 302
  - determining if locked, 302
  - determining if open, 304
  - drawing, 20
- picture data
  - locking, 304
  - unlocking, 304
- pie
  - drawing, 21, 473
  - from GraphWin, 473
- Pnt class, 490
- point, translating, 225
- pointer
  - to screen window, 11
  - to task window, 12
- polygon
  - drawing, 22, 473
  - from GraphWin, 473
- polyline
  - drawing, 23, 474
  - from GraphWin, 474
- polymorphism, 3
- position
  - moving, 472
  - retrieving for pen, 27
  - setting for pen, 32
  - setting for window, 148
- print record
  - retrieving, 313
  - retrieving size, 313
  - setting, 315
  - validating, 315
- print window, initializing, 314
- printer, 221
- printing, 312
- ProcessEvents
  - from XVT\_GlobalAPI, 204
- processing pending events, 204
- put\_def
  - from BaseXVT, 429
- PutData
  - from XVT\_ClipBoard, 78
- Q**
- QuitOK
  - from XVT\_TaskWin, 367
- quitting
  - application, 367
  - notifying object of request, 364
- R**
- radio button, 212
  - checking, 437
  - determining if checked, 441
  - setting, 320

- Raise, 167
  - from XVT\_AllocErrorManager, 9
  - from XVT\_ErrorManager, 170
  - from XVT\_InternalErrorManager, 237
- raising
  - error, 9, 170
  - internal error, 237
- Rct class, 493
- ReadAccess
  - from XVT\_GlobalAPI, 205
- rectangle
  - computing difference, 326
  - computing intersection, 328
  - constraining a point to, 324
  - determining contents, 325
  - determining if empty, 329, 494
  - drawing, 23, 474
  - from GraphWin, 474
  - retrieving, 12, 13, 426
  - retrieving dimensions, 327
  - retrieving height, 328
  - retrieving lower left, 326
  - retrieving lower right, 327
  - retrieving upper left, 327
  - retrieving upper right, 328
  - retrieving width, 332
  - setting dimensions, 495
  - setting lower left, 330
  - setting lower right, 330
  - setting upper left, 330
  - setting upper right, 331
- rectangular region, scrolling, 145
- red
  - retrieving color component, 82
  - setting color component, 83
- ReleaseMouse
  - from XVT\_ChildBase, 55
- releasing the mouse, 55
- rem
  - from StrList, 514
- Remove
  - from XVT\_StrList, 355
- removing
  - all items from list, 245
  - all items from list box, 443, 444
  - element from string list, 355, 514
  - item from list, 246
  - item from list box, 444
  - text from text edit object, 373
- Replace
  - from XVT\_Menu, 272
- replacing default menu item, 272
- resizing global memory, 198
- resource ID
  - retrieving, 85, 86
  - retrieving strings by, 201
  - setting, 87, 89
- Response
  - from XVT\_GlobalAPI, 206
- RestoreDir
  - from XVT\_GlobalAPI, 206
- restoring directory, 206
- resuming updates to list box, 255, 447
- retrieving
  - active text edit, 50
  - all items from list, 247
  - all selected items, 249
  - all selected items from list box, 446
  - application base name, 86
  - application configuration data, 520
  - application name, 85
  - ascent, 179
  - background color, 151, 191
  - blue color component, 81
  - bottom of rectangle, 494
  - brush, 26, 151, 460
  - brush color, 38, 434
  - brush pattern, 38, 434
  - caret position, 50
  - caret visibility, 51
  - client area, 418
  - client area boundary, 12
  - clipboard data, 76
  - clipping rectangle, 26

- clipping state, 27
- container event mask, 141
- contents of string list, 512
- contents of text line, 376
- control, 115
- control ID, 95, 441
- control object, 140
- control title, 239
- default directory, 192
- descent, 179
- dialog title, 118
- directory, 172, 194
- drawing mode, 27, 152, 461
- drawing tools, 28, 470
- element from string list, 353
- element in a string list, 513
- event delivery mask, 419
- event mask, 116
- file name, 172
- file type, 173
- first child window, 142, 334
- first control, 141
- first control in a dialog, 117
- first menu item, 270
- first selected item from list box, 248, 445
- font, 152, 460
- font family, 464
- font metrics, 28, 503
- font size, 176, 466
- font structure, 465
- font style, 466
- foreground color, 152
- frontmost top-level window, 191
- frontmost window, 192
- green color component, 81
- index of first selected item, 249
- index of first selected item in list box, 446
- item from list, 247
- leading, 180
- left edge of rectangle, 494
- menu, 287
- menu accelerator key, 278
- menu item, 270
- menu item tag, 484
- menu item title, 278
- next child window, 143, 334
- next control, 142
- next element in string list, 354, 514
- nonportable string directory specification, 126
- number of characters in text edit line, 378
- number of child windows, 143, 335
- number of controls, 140
- number of controls in dialog, 116
- number of elements in list, 511
- number of elements in string list, 352
- number of items in list, 245
- number of items in list box, 443
- number of lines in text edit object, 378
- number of lines in text edit paragraph, 379
- number of menu items, 270
- number of paragraphs in text edit object, 379
- number of selected items, 246
- number of selected items in list box, 444
- object title, 427
- offset to view rectangle, 380
- opaque text flag, 153
- outer rectangle, 13, 426
- output file, 312
- output file name, 311
- parent menu, 282
- parent window, 51, 95, 428
- pen, 28, 153, 461
- pen color, 296, 487
- pen pattern, 296, 487
- pen position, 27
- pen style, 296
- pen width, 297, 488
- print record, 313
- rectangle dimensions, 327
- rectangle height, 328
- rectangle lower left, 326
- rectangle lower right, 327
- rectangle upper left, 327

- rectangle upper right, 328
- rectangle width, 332
- red color component, 82
- resource ID, 85, 86
- right edge of rectangle, 495
- scrollbar position, 52
- scrollbar range, 53, 340, 441
- size of global block, 198
- size of print record, 313
- strings from resources, 195
- strings with consecutive resource IDs, 201
- submenu pointer, 357
- subsequent controls in a dialog, 117
- subsequent menu items, 271
- system attribute value, 190
- task window title, 87
- text edit object, 54
- text edit object attributes, 375
- text edit object border, 375
- text edit object character limit, 376
- text edit object font, 376
- text edit object margin, 377
- text edit selection, 380
- text string width, 29
- thumb position, 339
- thumb proportion, 52, 340, 440
- timer interval, 392
- toggle state, 394
- top of rectangle, 496
- user data for control, 196
- user data for dialog, 193
- user data for menu item, 194
- view rectangle, 381
- window definition, 426
- window handle, 419
- window title, 288
- window type, 13, 427
- x coordinate, 307, 491
- y coordinate, 307, 492
- returning to startup directory, 224
- right
  - from Rct, 495
- rounded rectangle, drawing, 24, 475
- rounded\_rectangle
  - from GraphWin, 475
- S**
- SaveDir
  - from XVT\_GlobalAPI, 207
- SaveFile
  - from XVT\_GlobalAPI, 207
- saving
  - directory, 207
  - file, 207
- SC\_\* (scroll control), 339
- screen, 222
- screen window, 11
- ScreenWin class, 3, 498
- Scroll, 138
  - from XVT\_BaseDrawProto, 145
- SCROLL\_CONTROL enumeration, 339
- scrollbar
  - initializing, 341
  - notifying object of activity, 49, 338
  - retrieving position, 52
- scrollbar range
  - retrieving, 53, 340, 441
  - setting, 60, 343, 450
- scrolling
  - rectangular region, 145
  - text edit, horizontally, 374
- select\_text
  - from Control, 449
- selecting
  - font, 465
  - item, 250
  - text, 449
- set
  - from Pnt, 491
  - from Rct, 408, 495
- set\_brush
  - from GraphWin, 475
- Set\_Checked\_State, 394

- set\_def
  - from DlgWin, 408, 455
  - from ScreenWin, 408, 503
- set\_font
  - from BaseWin, 419
  - from Font, 408, 465
  - from GraphWin, 476
- set\_mask
  - from BaseWin, 420
- set\_mode
  - from GraphWin, 476
- set\_pen
  - from GraphWin, 477
- set\_scroll\_pos
  - from Control, 449
- set\_scroll\_proportion
  - from Control, 450
- set\_scroll\_range
  - from Control, 450
- set\_text
  - from BaseXVT, 429
- set\_timer
  - from BaseWin, 408, 420
- set\_tools
  - from GraphWin, 477
- SetAboutBoxID
  - from XVT\_Config, 87, 183
- SetActive
  - from XVT\_TextEdit, 384
- SetAppName
  - from XVT\_Config, 88
- SetAscent
  - from XVT\_FontMetrics, 180
- SetAttrValue
  - from XVT\_GlobalAPI, 110, 131, 208
- SetBackColor
  - from XVT\_BaseDrawProto, 30
  - from XVT\_DrawTools, 153
- SetBaseAppName
  - from XVT\_Config, 88
- SetBlue
  - from XVT\_Color, 82
- SetBottomLeft
  - from XVT\_Rct, 330
- SetBottomRight
  - from XVT\_Rct, 330
- SetBrush
  - from XVT\_BaseDrawProto, 31
  - from XVT\_DrawTools, 154
- SetCaretDimensions
  - from XVT\_ChildBase, 56
- SetCaretPos
  - from XVT\_ChildBase, 56
- SetCaretState
  - from XVT\_ChildBase, 57
- SetCheckedState
  - from XVT\_CheckBox, 45
  - from XVT\_MenuItem, 275
  - from XVT\_RadioButton, 320
- SetClip
  - from XVT\_BaseDrawProto, 31
- SetClipState
  - from XVT\_BaseDrawProto, 32
- SetColor
  - from XVT\_Brush, 39
  - from XVT\_Pen, 297
- SetCurrentPoint
  - from XVT\_BaseDrawProto, 32
- SetCursor
  - from XVT\_ChildBase, 57
- SetDescent
  - from XVT\_FontMetrics, 180
- SetDir
  - from XVT\_FileSpec, 173
- SetDocTitle, 88, 402
  - from XVT\_MenuWin, 289
- SetDrawMode
  - from XVT\_BaseDrawProto, 33
- SetDrawTools
  - from XVT\_BaseDrawProto, 34

- SetEnabledState
  - from XVT\_ChildBase, 58
  - from XVT\_Control, 97
  - from XVT\_Dialog, 120
  - from XVT\_MenuNode, 279
- SetEventMask
  - from XVT\_Dialog, 120
  - from XVT\_DrawableContainer, 146
- SetFileType
  - from XVT\_GlobalAPI, 224
- SetFont
  - from XVT\_BaseDrawProto, 34
  - from XVT\_DrawTools, 154
- SetFontMenu
  - from XVT\_MenuWin, 289
- SetForeColor
  - from XVT\_BaseDrawProto, 35
  - from XVT\_DrawTools, 155
- SetGreen
  - from XVT\_Color, 82
- SetInnerRect, 114, 137
  - from XVT\_Control, 98
  - from XVT\_Dialog, 121
  - from XVT\_DrawableContainer, 148
- setjmp, 168
- SetLeading
  - from XVT\_FontMetrics, 181
- SetMenu
  - from XVT\_MenuWin, 290
- SetMenuBarID
  - from XVT\_Config, 89
- SetMode
  - from XVT\_DrawTools, 155
- SetName
  - from XVT\_FileSpec, 174
- SetOpaqueText
  - from XVT\_DrawTools, 26, 156
- SetOutputFile
  - from XVT\_PrintWin, 314
- SetPattern
  - from XVT\_Brush, 39
  - from XVT\_Pen, 297
- SetPen
  - from XVT\_BaseDrawProto, 36
  - from XVT\_DrawTools, 157
- SetPrintRecord
  - from XVT\_PrintWin, 315
- SetRed
  - from XVT\_Color, 83
- SetScrollPosition
  - from XVT\_ChildBase, 59
  - from XVT\_ScrollBar, 342
- SetScrollProportion
  - from XVT\_ChildBase, 60
  - from XVT\_ScrollBar, 342
- SetScrollRange
  - from XVT\_ChildBase, 60
  - from XVT\_ScrollBar, 343
- SetSelectedState
  - from XVT\_List, 250
- SetSize
  - from XVT\_Font, 176
- SetStyle
  - from XVT\_Pen, 298
- SetSubMenuPtr
  - from XVT\_SubMenu, 357
- SetSuspendedState
  - from XVT\_ListBox, 255
- SetTaskWinTitle
  - from XVT\_Config, 89
- setting
  - application base name, 88
  - application name, 88
  - ascent, 180
  - attribute value, 208
  - background color, 30, 153
  - blue color component, 82
  - brush, 31, 154, 460, 475
  - brush color, 39, 434
  - brush pattern, 39, 434
  - caret dimensions, 56
  - caret position, 56
  - caret visibility, 57
  - clipping region, 31



- clipping state, 32
- container event mask, 146
- control dimensions, 98
- control title, 240
- coordinates, 491
- descent, 180
- dialog position, 121
- dialog size, 121
- dialog title, 122
- directory, 173
- drawing mode, 33, 155, 461, 476
- drawing tools, 34, 477
- event delivery mask, 420
- event mask, 120
- file name, 174
- file type, 174, 224
- font, 34, 154, 419, 460, 476
- font family, 464
- font menu checkmarks, 289
- font size, 176, 466
- font style, 466
- foreground color, 35, 155
- green color component, 82
- leading, 181
- menu, 290
- menu item title, 279
- object title, 429
- opaque text flag, 156
- output file name, 314
- pen, 36, 157, 461, 477
- pen color, 297, 487
- pen pattern, 297, 487
- pen position, 32
- pen style, 298
- pen width, 299, 488
- print record, 315
- radio button, 320
- rectangle dimensions, 495
- rectangle lower left, 330
- rectangle lower right, 330
- rectangle upper left, 330
- rectangle upper right, 331

- red color component, 83
- resource ID, 87, 89
- scrollbar range, 60, 343, 450
- submenu pointer, 357
- task window title, 89
- thumb position, 59, 342, 440, 449
- thumb proportion, 60, 342, 450
- timer, 420
- WIN\_DEF structure, 455, 503
- window cursor shape, 57
- window definition, 429
- window position, 148
- window size, 148
- window title, 289, 291
- x coordinate, 308, 491
- y coordinate, 308, 492
- SetTitle, 402
  - from XVT\_Dialog, 122
  - from XVT\_Label, 240
  - from XVT\_MenuNode, 279
  - from XVT\_MenuWin, 291
- SetTopLeft
  - from XVT\_Rct, 330
- SetTopRight
  - from XVT\_Rct, 331
- SetType
  - from XVT\_FileSpec, 174
- SetVisibleState
  - from XVT\_ChildBase, 61
  - from XVT\_Control, 98
  - from XVT\_Dialog, 123
- SetWidth
  - from XVT\_Pen, 299
- SetX
  - from XVT\_Pnt, 308
- SetY
  - from XVT\_Pnt, 308
- show
  - from BaseXVT, 430
- showing an object, 430
- SHRT\_MAX, 59, 61
- SHRT\_MIN, 59, 61

- size
  - from Font, 466
  - notifying object of change, 113, 136
  - setting for window, 148
- SLIST, 445, 446
- source file, vi
- starting an XVT++ application, 520
- startup directory, returning to, 224
- StartupDir
  - from XVT\_GlobalAPI, 224
- strcat, 75
- string
  - getting from user, 206
  - retrieving, 195, 201
  - text, 25, 478
- string list
  - adding items, 350
  - adding to in order, 351, 511
  - appending to debug file, 352
  - determining if valid, 515
  - removing element, 355, 514
  - retrieving contents, 512
  - retrieving element, 353, 513
  - retrieving next element, 354, 514
  - retrieving number of elements, 352
  - traversing, 353, 513
  - writing to debug file, 512
- StrList class, 509
- StrListElt class, 517
- style
  - from Font, 466
  - retrieving for pen, 296
  - setting for pen, 298
- subclass, vi
- submenu
  - retrieving pointer, 357
  - setting pointer, 357
- superclass, vi
- suspending updates to list box, 255, 448
- switch statement, 2
- system attribute
  - retrieving value, 190
  - setting, 208
- T**
- tag
  - from MenuItem, 484
- task window, 12
  - initializing, 366
  - retrieving title, 87
  - setting title, 89
- TaskWin class, 518
- terminating, 187
- text
  - from GraphWin, 478
  - selecting, 449
- text edit
  - attribute flags, 382
  - retrieving, 50
  - retrieving selection, 380
  - scrolling horizontally, 374
- text edit object
  - adding paragraph, 371
  - deleting paragraph, 373
  - initializing, 382
  - removing text, 373
  - retrieving, 54
  - retrieving attributes, 375
  - retrieving border, 375
  - retrieving character limit, 376
  - retrieving contents of line, 376
  - retrieving font, 376
  - retrieving margin, 377
  - retrieving number of characters in line, 378
  - retrieving number of lines, 378
  - retrieving number of paragraphs, 379
- text flag
  - retrieving, 153
  - setting, 156
- text string
  - drawing, 25, 478
  - retrieving width, 29

- thumb position
  - retrieving, 339
  - setting, 59, 342, 440, 449
- thumb proportion
  - retrieving, 52, 340, 440
  - setting, 60, 342, 450
- timer
  - notifying object of expiration, 114, 137
  - retrieving interval, 392
  - setting, 420
- title
  - retrieving, 427
  - retrieving for control, 239
  - retrieving for dialog, 118
  - retrieving for menu item, 278
  - retrieving for window, 288
  - setting, 429
  - setting for control, 240
  - setting for dialog, 122
  - setting for menu item, 279
  - setting for window, 289, 291
- title bar, 223
- toggle
  - notifying object of activity, 394
  - retrieving state, 394
- tools.h, 81
- top
  - from Rct, 496
- top-level window
  - initializing, 401
  - retrieving, 191
- TranslatePoints
  - from XVT\_GlobalAPI, 225
- translating
  - point, 225
  - point relative to origin, 332
  - point relative to rectangle, 331
- TransToGlobal
  - from XVT\_Rct, 331
- TransToLocal
  - from XVT\_Rct, 332
- TrapMouse
  - from XVT\_ChildBase, 62
- trapping the mouse, 62
- traversing a string list, 353, 513
- TX\_\* flags, 382
- TX\_ONEPAR, 376
- typefaces, listing, 200
- typesetting conventions, v
- U**
- UCHAR\_MAX, 110
- UCHAR\_MAX constant, 110, 131
- uncheck
  - from Control, 451
  - from MenuItem, 484
- unchecking
  - check box, 45
  - checkbox, 451
  - menu item, 275, 483, 484
- Unlock
  - from XVT\_Picture, 304
- unlocking
  - global memory, 199
  - picture data, 304
- unselecting an item, 250
- UpdateWindow, 139
  - from XVT\_BaseDrawProto, 36
- updating
  - list box, 447
  - window, 36
- url.h, 183
- user
  - asking a question of, 183
  - getting a string from, 206
- user-defined cursor, 58
- user-defined event, 94, 115
- using XVT++, 1
- V**
- valid
  - from StrList, 408, 515
- ValidatePrintRcd, 204
  - from XVT\_PrintWin, 315

validating print record, 315  
 vertical scrollbar, notifying object of activity, 49  
 view rectangle  
   retrieving, 381  
   retrieving offset, 380  
 visibility  
   determining for control, 96  
   determining for dialog, 118  
   determining for window, 54  
   retrieving for caret, 51  
   setting for caret, 57  
   setting for control, 98  
   setting for dialog, 123  
   setting for window, 61

## W

W\_DBL, 102, 401  
 W\_DOC, 102, 401  
 W\_NO\_BORDER, 67  
 W\_PLAIN, 67, 102, 401  
 WD\_MODAL, 454  
 WD\_MODELESS, 454  
 Width  
   from XVT\_Rct, 332  
 width  
   from Pen, 488  
   retrieving for pen, 297, 488  
   retrieving for rectangle, 332  
   retrieving for text string, 29  
   setting for pen, 299, 488  
 WIN\_DEF structure, 439  
   creating control from, 438  
   creating window from, 501  
   setting, 455, 503

## window

  clearing, 129, 470  
   creating, 500, 501, 502  
   destroying, 129  
   determining if area needs drawing, 30  
   determining visibility, 54  
   determining whether enabled, 51  
   disabling, 58  
   drawing, 128

  enabling, 58  
   giving focus, 55  
   initializing, 102  
   initializing child, 67  
   initializing top level, 401  
   moving to front, 55  
   notify object of invalidation, 137  
   retrieving caret position, 50  
   retrieving child, 334  
   retrieving definition, 426  
   retrieving first child, 142  
   retrieving frontmost, 192  
   retrieving handle, 419  
   retrieving next child, 143, 334  
   retrieving number of children, 143, 335  
   retrieving parent, 51, 95  
   retrieving title, 288  
   retrieving top level, 191  
   retrieving type, 13, 427  
   setting caret dimensions, 56  
   setting caret position, 56  
   setting cursor shape, 57  
   setting definition, 429  
   setting position, 148  
   setting size, 148  
   setting title, 289, 291  
   setting visibility, 57, 61  
   updating, 36

Window, native X, 220

Windowptr, 220

writing string list to debug file, 512

WSF\_HSCROLL, 212

WSF\_VSCROLL, 213

## X

x

  from Pnt, 491

x coordinate

  retrieving, 307, 491

  setting, 308, 491

XVT Portability Toolkit, 1

XVT/CH, 16, 17, 18, 19, 20, 21, 22, 23, 25, 35, 56,  
   58, 110, 131, 155, 204, 209, 303, 313,

- 315, 342, 343
- XVT/Mac, 18, 20, 55, 110, 131, 189, 203, 205, 208, 209, 224, 303, 316, 343
- XVT/PM, 18, 20, 101, 110, 131, 144, 205, 209, 210, 303, 314, 316, 366
- XVT/Win, 18, 20, 55, 101, 110, 131, 144, 190, 205, 209, 303, 316, 366
- XVT/XM, 18, 122, 148, 204, 210, 313, 315
- XVT/XOL, 20, 204, 210, 313, 315
- XVT\_AllocError global variable, 8
- XVT\_AllocErrorHandler class, 5
- XVT\_AllocErrorManager class, 8
- XVT\_Base class, 10
- XVT\_BaseDrawProto class, 15
- XVT\_Brush class, 37
- XVT\_Button class, 41, 97
- XVT\_CB global variable, 72
- XVT\_CheckBox class, 44, 97
- XVT\_ChildBase class, 48
- XVT\_ChildWin class, 66
- XVT\_ClipBoard class, 72
- XVT\_Color class, 80
- XVT\_Color object, 80, 191
- XVT\_Config class, 84
- XVT\_Config instance, 200, 289
- XVT\_Config structure, 183
- XVT\_Container class, 90
- XVT\_Control class, 92
- XVT\_DetachedWin class, 1, 101, 498
- XVT\_Dialog class, 1, 108
- XVT\_Directory class, 125
- XVT\_Directory instance, 171
- XVT\_DrawableContainer class, 3, 127, 391
- XVT\_DrawTools class, 37, 150, 459
- XVT\_Edit class, 158
- XVT\_Editable class, 161
- XVT\_ErrorHandler class, 166
- XVT\_ErrorManager class, 166, 169
- XVT\_FileSpec class, 171
- XVT\_Font class, 175, 463
- XVT\_FontMetrics class, 178
- XVT\_GlobalAPI class, 182
- XVT\_GroupBox class, 226
- XVT\_Icon class, 229
- XVT\_INTERNAL\_ERROR macro, 237
- XVT\_InternalError global variable, 236
- XVT\_InternalErrorHandler class, 233
- XVT\_InternalErrorManager class, 236
- XVT\_Label class, 238, 346
- XVT\_List class, 243
- XVT\_ListBox class, 253
- XVT\_ListButton class, 258
- XVT\_ListEdit class, 262
- xvt\_malloc function, 218
- XVT\_Menu class, 267, 482
- XVT\_Menu structure, 267
- XVT\_MenuItem class, 273, 482
- XVT\_MenuNode class, 277
- XVT\_MenuNodeBase class, 281
- XVT\_MenuSeparator class, 283
- XVT\_MenuWin class, 285
- XVT\_Pen class, 295, 486
- XVT\_Picture class, 300
- XVT\_Picture type, 75
- XVT\_Pnt class, 306, 490
- XVT\_PrintWin class, 309
- XVT\_RadioButtonGroup instance, 318
- XVT\_RadioButton class, 318
- XVT\_Rct class, 323, 493
- xvt\_realloc function, 218
- XVT\_ScreenWin class, 333
- XVT\_ScrollBar class, 337
- XVT\_StrList, 509
- XVT\_StrList class, 517
- XVT\_Strlist class, 349
- XVT\_SubMenu class, 356
- XVT\_TaskWin class, 1, 359, 518
- XVT\_TextEdit class, 370
- XVT\_Timer class, 390
- XVT\_Timer instance, 137
- XVT\_Toggle class, 393
- XVT\_TopLevelWin class, 101, 397, 498
- XVT\_ToplevelWin class, 1
- XVTDEBUG file, 186

## **Y**

y

from Pnt, 492

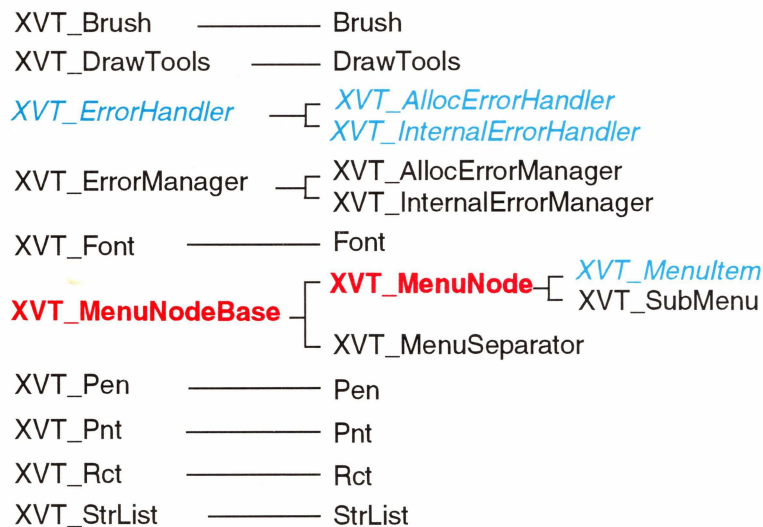
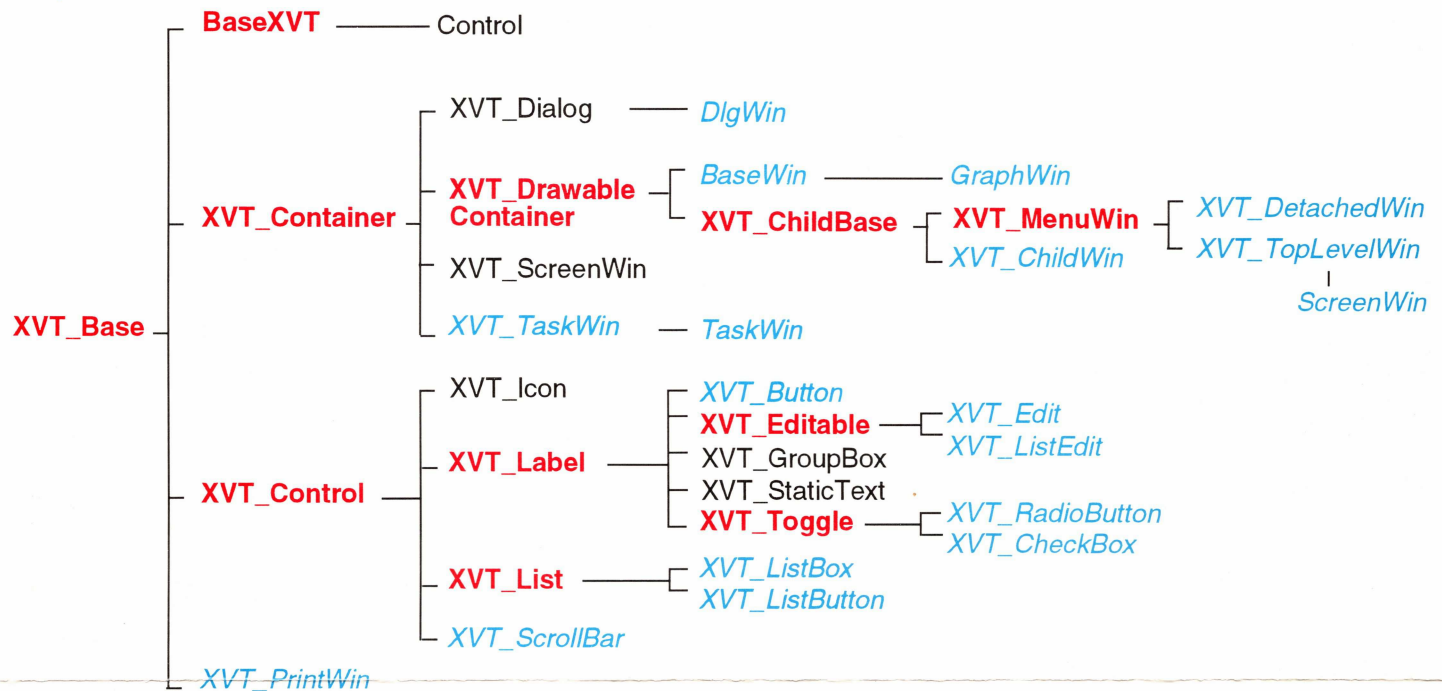
y coordinate

retrieving, 307, 492

setting, 308, 491, 492



# XVT++ 2.0 Class Hierarchy



## Standalone Classes

MenuItem  
 StrListElt  
**XVT\_BaseDrawProto**  
 XVT\_ClipBoard  
 XVT\_Color  
 XVT\_Config  
 XVT\_Directory  
 XVT\_FileSpec  
 XVT\_FontMetrics  
 XVT\_GlobalAPI  
 XVT\_Menu  
 XVT\_Picture  
 XVT\_RadioBtnGroup  
 XVT\_TextEdit  
 XVT\_Timer

## Key

Concrete Class

Abstract Class

Implementation Class